# An Effective Simulator for Hardware-In-Loop Simulation Verification of Truck platooning

Mariappan Haribalasubramaniam, Sandeep

*Award date:*
2020

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

# Declaration concerning the TU/e Code of Scientific Conduct for the Master's thesis

I have read the TU/e Code of Scientific Conduct[i].

I hereby declare that my Master's thesis has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct

Date

12-08-2020

Name

SANDEEP MARIAPPAN HARIBALASUBRAMANIAM

ID-number

1351885

Signature

M.H. Sandy

*Submit the signed declaration to the student administration of your department.*

[i] See: https://www.tue.nl/en/our-university/about-the-university/organization/integrity/scientific-integrity/
The Netherlands Code of Conduct for Scientific Integrity, endorsed by 6 umbrella organizations, including the VSNU, can be found here also. More information about scientific integrity is published on the websites of TU/e and VSNU

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science
System Architecture and Networking Group

# An Effective Simulator for Hardware-In-Loop Simulation Verification of Truck platooning

Sandeep Mariappan Haribalasubramaniam

Supervisors:
Dr. Ion Barosan(Tu/e)
Ihsan Yalcinkaya(TNO)

A thesis submitted in the partial fulfillment of the requirements of the degree
Master of Science in Embedded Systems

Eindhoven, August, 2020

# Abstract

Recent developments in the Autonomous driving and in Advance Driver Assistance Systems (ADAS) have increased the research interests in the Truck Platooning application. The benefits the truck platooning provide include, but not limited to, increased fuel efficiency, decreased CO2 emission, improvements in road traffic etc. These widespread benefits of the truck platooning demands the application to become practically viable. However, the truck platooning involves a high degree of autonomy and there are scenarios where the vehicles need to interact with vehicles that are in the platoon and in some instances also with other vehicles in the road traffic. These complex autonomous scenarios and the high degree of autonomy carries more risks when utilised in real world traffic. Hence guaranteeing reliability and safety is a major concern with truck platooning. Therefore, a proper verification methodology is required to ensure safety of the system. Though there are several verification methodologies, the focus of this thesis is on Hardware-In-Loop (HIL) simulation verification of truck platooning and addressing one of the major challenges in the HIL verification of truck platooning. HIL testing makes use of simulators to emulate the actual behaviour of the sensors and actuators and the actual Electronic Control Unit (ECU) will be embedded in the testbed. Hence, the choice of simulator is a major challenge because it not only should have the capabilities to emulate the real world sensors and actuators but in addition should also have the capability to simulate the environment and the interaction with vehicles such that it is close to reality. To address this challenge, the thesis proposes a methodology to evaluate a simulator for HIL verification of Truck platooning and consequently proposes an effective simulator for HIL verification of truck platooning.

Firstly, there is a need for criteria or parameters that the simulator should possess based on which the simulators can be evaluated. Therefore, first a thorough literature survey was done to determine and define the Key Performance Indicators (KPIs) for the simulators. These KPIs are determined taking into account the general requirements that an autonomous vehicle simulator should possess, the truck platooning application requirements and also some KPIs that the HIL setup demands. After determining a number of KPIs , depending on the type of application and depending on the project needs, the impact and weights of KPIs need to be determined. This plays a key role in the evaluation of simulators since not all the KPIs have the same importance in the context of the application. Hence, a survey was done with domain experts to find the weights. A technique called Analytic hierarchy Process was used which is a well-known Multi Criteria Decision Making technique to find the weights.

Subsequently, after finding the weights, there should be a method to validate the simulators based on the KPIs. Scenario based testing is a common testing method which is used in HIL. Therefore, it was decided to use a common truck platooning scenario to validate the KPIs and find an effective simulator. The decided Use Case is the Cutin scenario in front of the lead vehicle in the platoon. This scenario covers the important KPIs that are found during the weight analysis. Consequently, to find the effective simulator the scenario is divided into components and the KPIs are mapped to the components. Moreover, this enables to define a scoring system for each of the KPI and for each simulator. Overall, based on the evaluation, the end result is a score for the simulators based on the KPIs from which the effective simulator can be found. This result can be easily extended to find the simulator for the autonomous vehicle projects in which a different KPI could be of more importance.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

There have been plenty of developments, in the past decade, in the field of autonomous vehicles. Autonomous driving has sparked enormous interests in the automotive research field and the number of safety systems and driver assist systems are in an increasing trend[8]. Some of these latest technologies include, but not limited to, Adaptive Cruise Control (ACC)[9], Co-operative Adaptive Cruise Control (CACC) [10], Autonomous Emergency breaking(AEB) [11] etc. Moreover, advancements in Vehicle-to-Vehicle communication are made which enables transfer of crucial information between different vehicles. Combining all these advanced driver assist technologies paves way for making the autonomous vehicle a reality. Autonomous vehicles also play a major role in mitigating the number of accidents that occur due to human driver errors [12]. However, reliability is an indispensable factor in autonomous driving. It is estimated that an autonomous vehicle is considered safe and reliable after it has run 11 billion miles according to Nidhi et al [13].

It is not realistically possible to guarantee reliability based on the above mentioned statistic. However, effective verification methodologies of these autonomous systems can improve reliability. The problem with these advanced autonomous driving technologies is in testing them effectively. These autonomous vehicles must be tested on isolated highways and progressively with various real traffic scenarios. However, testing them in real traffic scenarios is not completely feasible because any failure could lead to a fatal situation. This is a growing concern in the field of autonomous driving. One possible method is to simulate the real-world traffic scenarios and observe the behaviour and make the changes in the system accordingly.

Model Based testing of the automotive systems is an effective method of testing these safety-critical systems. Some of these testing methodologies are the Model-in-Loop testing (MIL), Software-in-Loop testing (SIL) and Hardware-In-Loop testing (HIL)[14]. It can be observed based on the description in [14] that the HIL is more effective than the other strategies of model based testing of automotive systems. This is more valid, especially, in case of autonomous vehicles. Because in HIL the actual Engine Control unit (ECU) is integrated within the test environment. Thus, the real-time behaviour of the model can be observed and the behaviour would be the same as in the real application. Since the HIL simulation emulates the actual vehicle behaviour, the HIL methodology is an effective verification strategy for the autonomous vehicles. HIL is not only an effective in autonomous/automotive vehicles. It is also used in verification of other critical systems which is discussed in Section 2.1.

Although in HIL the ECU is integrated in the test environment, the actual environment around the ECU is a simulated environment. Therefore, a simulator is necessary to create the real world environment and generate real world traffic. The choice of the simulator depends on the simulator capabilities such as to create a 3-D model of the actual world and emulate traffic scenarios. Usually robotic simulators are used for the HIL testing since they are handy in creating 3-D world models. Moreover, simulators such as LGSVL simulator have several sensors and environmental models

[15] which is conducive to emulate the actual world environment in the simulator. Such features make the robotic simulators an effective and appropriate tool which can be used in HIL to make the verification of the autonomous vehicles convenient. The robotic simulators also promote the notion of Digital twins [16]. Digital twins are the virtual substitutes of real world objects consisting of real world virtual representations and communication services as mentioned in [16]. It is also described in [16] that the digital twins break new ground in simulation-based development and operation of complex technical systems. The digital twins not only help in the development phase but also in verification of those systems which will be explored in this dissertation. Hence, Simulator constitute a vital part in the HIL based verification.

On the other hand, extensive growth in the autonomous driving technologies pave way for several novel techniques in autonomous vehicles which could be implemented in real-time traffic. One such emerging and most promising concept is Truck platooning. Truck platooning is a concept where trucks are aligned into a homogeneous group with minimal distance between each member-vehicle[17]. This is enabled by Vehicle-to-Vehicle communication and few other autonomous driving technologies such as Co-operative Adaptive Cruise Control (CACC), Collision Avoidance (CA), Collision Warning (CW) etc. Platooning of vehicles is shown to have several benefits in literature. Since Vehicles are close to each other the road capacity can be increased [18]. Moreover, the platoon can reduce energy consumption and reduced CO2 emissions because of the streamlining of vehicles which can minimize air drag [19, 18].

Given all the above mentioned concepts, this thesis focuses on analyzing and finding an effective simulator for HIL based testing platform for autonomous vehicles. In detail this dissertation focuses mainly on the various simulators and their impact in HIL testing of autonomous vehicles with major emphasis on the Truck Platooning application. The effect of simulation engines in creating the 3-D representation of the real world and their ease of usage will be explored. The HIL setup which is already used in TNO will be utilised for the validating and benchmarking the simulators against the truck platooning application. The Key Performance Indicators that are necessary for a simulator for the HIL verification of Truck platooning will be defined and their impact will be weighed. The scenario for the truck platooning application which will be used for the evaluation of the simulators will be identified. The scenario will be created in the simulation environment and behaviour will be verified. The implementation and performance capabilities of the simulators will be analysed based on the KPIs and the simulators will be evaluated. Finally, this thesis proposes a simulator for HIL verification for truck platooning application for the HIL setup in TNO which can be easily extended to any HIL setup and verification of any complex automotive system.

## 1.1 Problem Statement

This thesis aims at proposing a method to find an effective simulator and find an effective simulator for a truck platooning application given a number of available simulators.

It is briefly explained above the importance of the Simulators in HIL testing. The HIL setup is a test bench that allows to test the (embedded) systems in a controlled closed loop with the emulation of sensors and actuators. The environment around the embedded system is emulated by the simulator software. Hence, there is a need for a powerful simulation tool capable of creating the environment and emulate the sensors and actuators around the embedded system. Therefore, there is a need for exploring and analysing various simulators that are suitable for the HIL setup and also for the truck platooning Use Cases. These simulators must also be compatible with the HIL setup in the TNO since these simulators will be evaluated on this setup i.e the simulators should be compatible with the Robot Operating System (ROS) middleware. Moreover, the truck platooning system is a very complex system consisting of several scenarios and Use Cases and a

large number of sensors and actuators.

The extent of the complexity of the truck platooning Use Cases vary some simple to highly complex. A Truck platooning scenario could be a truck wanting to join a platoon or leave a platoon or Cutin scenario by a car in between the platoon etc. A simple truck platooning scenario could be a truck joining a platoon, which is a simple Use Case with respect to Truck platooning which could become complex if the number of trucks increases, for example 8 or 9. A small increase in the number of trucks could increase the complexity of the Use Case multifold and hence the simulation complexity also increases. It is relatively simple to simulate 2 trucks for the above scenario in a simulator. But if the number of truck is increased to say 4 or 5 then several performance parameters of the simulator needs to be taken into account and even this Use Case becomes a complex. The increase in complexity speaks volumes about the importance for an effective simulator for this application. Hence, a proper Use Case scenario needs to be identified where the simulators performance can be effectively verified. The Use Case should be chosen that the main steps are common to all the simulators and only the underlying implementation details would vary which will help in the evaluation of the simulators.

The identification of an effective and appropriate simulator contributes hugely towards an effective HIL testing. The performance of the embedded system in HIL is equivalent to its real-time performance. The simulator, therefore, should also be able to emulate the real-time performances of the sensors and actuators. The simulator should be capable enough to simulate the real traffic scenarios, at the same time consuming less load. As indicated before the middleware used for communication between various processes is ROS. The simulator should be compatible with ROS middleware to be compatible with the HIL setup in the TNO. Although, most of the available simulators are compatible with ROS. Having specified the goal of this dissertation, the main research question is formulated as follows:

**Principal RQ :** ***What is an effective simulator for truck platooning application in a Hardware-In-Loop simulation based testing platform with ROS middleware and How can an effective simulator be found given a number of available simulators?***

As mentioned above, with the increase in complexity of the system there are several performance parameters of the simulation tools that needs to be taken into account for the efficient simulation of the environment. Therefore, the Key Performance Indicators (KPIs) for the simulators needs to be identified as a first step. The Key Performance Indicators may vary according to the needs of the application or the project. Key Performance Indicators should be determined on the basis of what is being simulated and what the application demands. Thus, the KPIs can be generic for any autonomous vehicle simulations and the truck platooning application demands certain KPIs on its own. Therefore, both these cases have to be taken into consideration while determining the Key Performance Indicators for the simulators. This determination of KPI for both generic and truck platooning specific makes this thesis easily extendable for usage of the simulator in autonomous vehicle simulation and for digital twins. The KPI determination is really crucial to this dissertation since based on these performance indicators the simulators will be benchmarked and evaluated.

It is to be noted that the segregation factors described in the Figure 1.1 are the factors with which the KPIs will be determined and this is not a classification of the KPIs. This determination of the Key Performance Indicators leads to the first research question which is formulated as follows:

**RQ1:** ***What are the Key Performance Indicators for an autonomous vehicle simulation engine and what is their impact in the context of HIL based testing and Truck Platooning?***

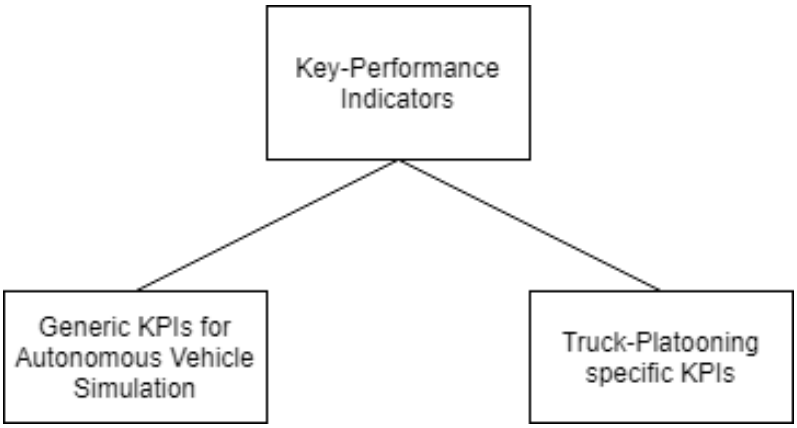


Figure 1.1: Key-Performance Indicator Segregation Factors

The answer to this research question not only determines the Key Performance Indicators but also answers the impact of the Key performance Indicators. This impact is quantified in the form of weights. Therefore, weight determination of the Key Performance Indicators will be a part of this research question. Though all the Key Performance Indicators are important for the simulators, some KPIs will be crucial for the specific application or indispensable in general. Hence, assigning weights to the KPIs is an important part of this research which itself calls for a sub-research question. This leads to a sub-research question which is formulated as follows:

**SUB RQ1:** ***How can the weights be determined for each of the Key Performance Indicator for the evaluation of simulators?***

With the answers to the above research questions, a solid basis is set for the benchmarking and evaluation of the simulation tools. However, this is still the first part of the method to find an effective simulator which the dissertation is focused. The second part is focused on the simulators that are to be compared, the quantification of the KPIs for the simulator by evaluating them with respect to the Use Case of truck platooning and the implementation of the Use Case in all the selected simulators.

Since the goal of this thesis is to find an effective simulator, comparisons have to be done between different simulators such that finally an appropriate and efficient simulator can be chosen for the intended HIL testbed. The basis on which the comparisons will be done is answered by the *RQ1*. The next step is evaluating the identified simulators. The identified Truck platooning Use Case will be executed on all the simulators and the performance comparisons will be made. Then the performance measures should be quantified and measured. This leads to the second research question which can be formulated as :

**RQ2:** ***How the Key Performance Indicators can be validated with respect to Truck Platooning application for different simulators?***

To validate the simulators a Use Case is necessary based on which the simulators will be evaluated. The Use Case which is considered for the evaluation is the Cutin scenario in front of the lead vehicle in the platoon with two vehicles. Cutin scenario is essentially the situation where two or more trucks are already engaged in a platoon i.e they are already in platooning state and follow each other in close distances. A Cutin happens when a non platoon vehicle which are the other road participants perform a manoeuvre and come in close proximity with the lead truck . When this happens, the lead vehicle has to adapt its speed otherwise it will collide with the vehicle which performs the manoeuvre. Since the lead vehicle reduces its speed, it will come in

close proximity with the trailing platoon vehicle and the trailing vehicle should also adapt its speed accordingly to maintain the platoon distance. This scenario is a common occurrence in Truck platooning application and this scenario is considered for the Simulator evaluation which is explained in detail in Chapter 4

## 1.2 Thesis Outline

The remainder of this thesis is structured as described in the Figure 1.2. Chapter 2 presents the background concepts that introduces certain terminologies and provides basic understanding about HIL, simulators, Scenario based testing and the selected simulators for analysis. This chapter also presents the related work. Chapter 3 explains the Key performance indicators and the determination of weights of the KPIs and answers the RQ1 and Sub RQ1. Chapter 4 details the implementation of the selected scenario in the three simulators and the implementation results. Chapter 5 presents the analysis and evaluation of the simulators based on the implementation of the Use Case. Chapters 4 and 5 together answer the RQ2. Finally, Chapter 6 concludes the dissertation and provides some future research directions.

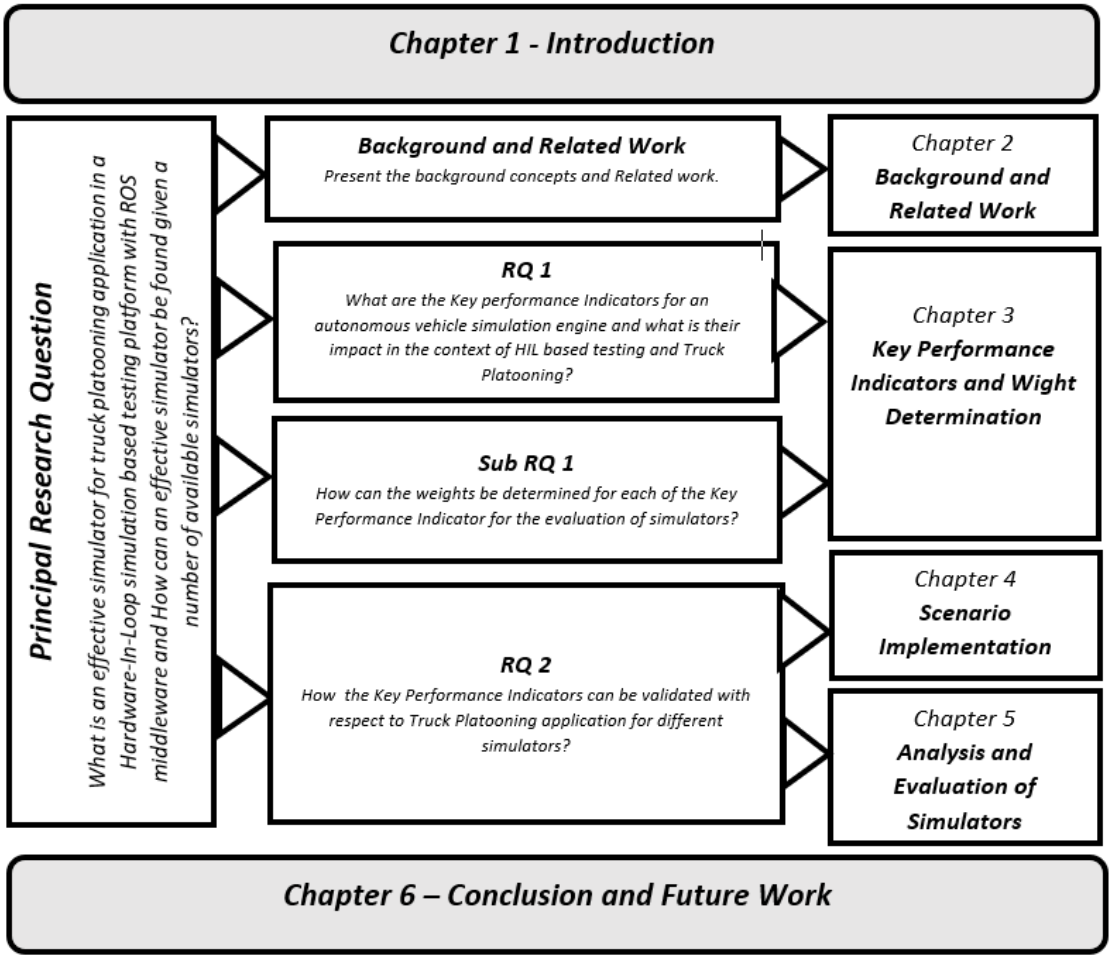


Figure 1.2: Thesis Outline

# Chapter 2

# Background and Related Work

This chapter gives information about the background concepts which would be helpful to understand the remaining chapters of this thesis. Initially, a brief introduction of the Hardware-In-Loop testing is provided which is followed by the Truck platooning application which is the application under consideration. Subsequently, a brief explanation of scenario based tesing is provided which is followed by the simulators that are selected for the evaluation to find the effective simulator. Additionally, a brief explanation of the TNO HIL setup is provided which explains the HIL testbed. The chapter is concluded with the related work that explains the similar works in the literature which are in the domain of this thesis.

## 2.1 Hardware-In-Loop Testing

Hardware-In-Loop simulation is one of the several testing methodologies [14]. It is not always possible to verify complex technical systems and safety-critical systems in real time. The first reason is that it is difficult to reproduce certain real-time scenarios and few futuristic scenarios. Moreover, testing of the safety-critical system in real time could be fatal. For example, consider a self driving car being tested on a highway or in urban-traffic. If there is any fault, it could lead to crashes or collision with the usual traffic which could even be fatal. Another reason is cost-effectiveness. The cost-effectiveness is a crucial factor since these complex systems are very difficult and expensive to manufacture and a major fault could damage the entire system. Also, the conventional testing methodologies are also slow to implement, time consuming and inflexible [20]. Hence, to effectively verify these systems there is a requirement of a methodology where the actual hardware is integrated but the environment in which the hardware is going to put to use is simulated. Hardware-in-Loop simulation testing facilitates the methodology. In HIL the actual embedded system is integrated in the testing but the environment around the embedded system is emulated by the simulator software. Hence, it is essential to make use of this testing mechanism in safety-critical domains such as Automotive and Aerospace domains. The main objective of the HIL, as mentioned in [14], is to find low-level ECU and I/O services faults. This testing mechanism is mainly used in System testing which is the penultimate stage of V-model of the software development [1].

Figure 2.2 depicts the various model-based testing strategies that are of best use in each stages of the software development. As observed from the Figure 2.2 HIL simulation testing is used in the penultimate stage of the V-model.

There are certain complexities associated with the HIL testing. Because of the interactions between the real time and the simulated systems, the simulated systems should be able to read the signals that are to be measured, perform real-time calculation and integration and should be able to output the results [2]. According to [2] HIL simulation helps in giving ECU an environment

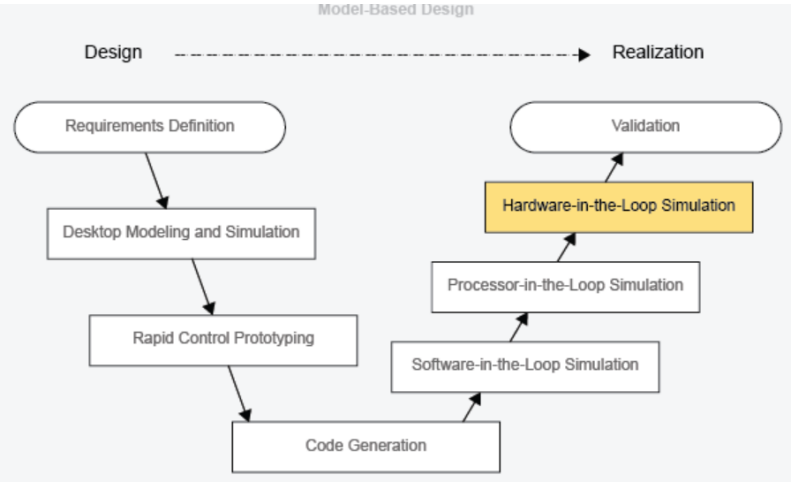


Figure 2.1: HIL simulation in V-Model [1]

as close to reality such that the ECU does not detect any inconsistencies in the electrical signal quality. Hence, any test that could be performed on the actual vehicle could also be performed in the HIL simulation. Therefore, some failure to meet real-time conditions could lead to unstable simulation and may also result in damage of the hardware.

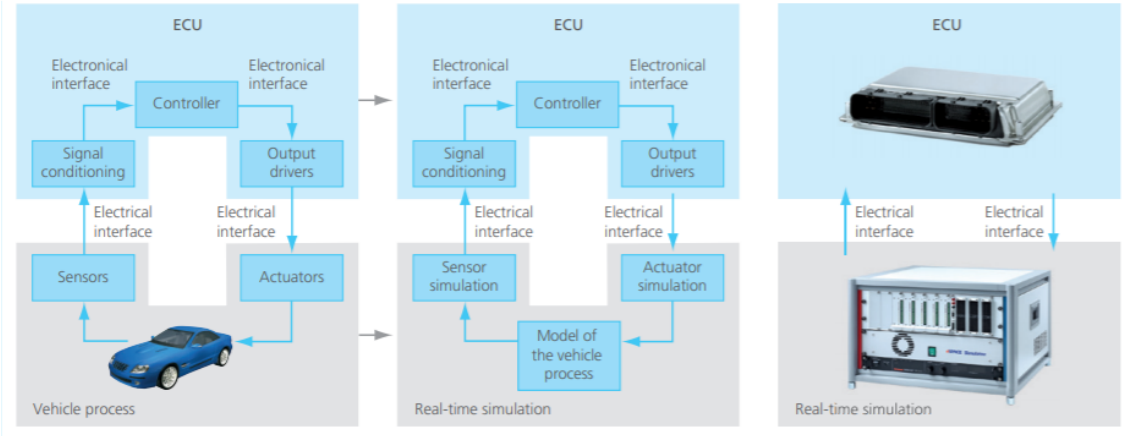


Figure 2.2: Signal flow in actual system and HIL simulation [2]

The Figure 2.2 illustrates the signal flows in an actual system and in a HIL simulation. As can be observed in Figure 2.2, in HIL testing the sensors, actuators and the vehicle are replaced by the sensor simulation, actuator simulation and model of the vehicle. The behaviour of the sensors and actuators are simulated which should emulate the real-time behaviour. The test scenarios are provided to the the hardware and the result is observed in the simulation environment. Hence, it is notable that the simulation environment and simulators play a major role in HIL and at the same time they should meet the demands of the application.

### 2.1.1 Basic components of a HIL System

The primary components of a HIL system constitutes the following components [3] and also can be observed in the Figure 2.3


Figure 2.3: Components of a HIL simulation[3]

**Real-Time Processor:** The real-time processor is the key component of the HIL simulation [3]. This is also where the simulation tool is integrated. The main purpose of this component is to execute the vehicle model, data logging and provide the stimulus.

**I/O Interfaces:** The I/O interfaces are mainly used for the interaction between the actual hardware and the simulated environment. Moreover, the test scenarios, acquisition of data logging are facilitated by the I/O interfaces.

**Operator Interface:** This is used for visualisation and for providing the test scenario. Also, this provides configuration management and test automation.

**Electronic Control Unit:** The ECU here is the actual hardware in which the software is integrated. Also, it can be noted that more than one ECUs can also be integrated in the HIL test set-up.

## 2.2 Truck Platooning

As mentioned in Chapter 1, the scenarios and Use Cases pertaining to Truck platooning will be used for the evaluation and benchmarking of the simulation tools. Essentially, Platooning is a concept where a vehicle closely and autonomously follows the vehicle in front forming platoons or convoys. The first vehicle, called the lead Vehicle, leads from the front and is usually assisted by a human driver [21]. The trailing vehicles may or may not be assisted by human drivers, although it is considered safe to have a human driver in the vehicle. This platooning is enabled by the efficient usage of appropriate sensors such as Radar, Camera, GPS etc and appropriate actuators and also with Vehicle-to-Vehicle (V2V) communication through which the autonomous vehicles communicate with each other and know each other's status constantly.

The benefits of Truck platooning are plenty. Since the vehicles are moving in traffic with very short distances between them, for instance 1 second or lesser than 1s, thanks to the vehicle-to-Vehicle communication, the road capacity will be increased. This contributes a lot to the reduction of the traffic and an increase in the throughput of the roadways. Moreover, since vehicles move in such short distances the air drag is decreased. This reduction in air drag is more significant in the truck platooning than for other vehicle platooning since the air drag experienced by the truck is relatively higher than a car or other vehicles [22]. This reduction in air drag in turn reduces the fuel consumption. As a result, truck platooning paves way for decreased fuel consumption of the vehicles. Therefore the benefits of truck platooning is not only limited to the vehicle itself but also for the environment.

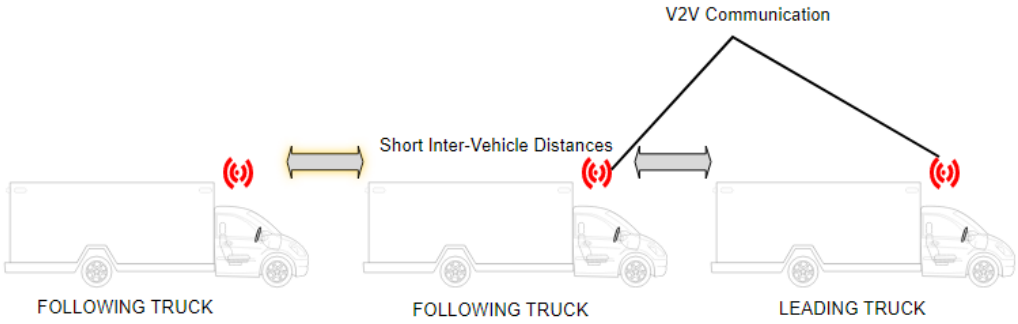


Figure 2.4: A simple Truck Platooning example

There are quite a number of ways in which Truck platooning can be designed and implemented. One of the interesting concept of the Truck platooning is the one by the ENSEMBLE (ENabling SafE Multi-Brand pLatooning for Europe). They had created a layered concept of Truck Platooning with each layers handling different functionalities. The software which will be used in this thesis for Truck Platooning will also be based on this ENSEMBLE project. The layered architecture of the Truck Platooning as illustrated in [4] is depicted in the Figure 2.5:

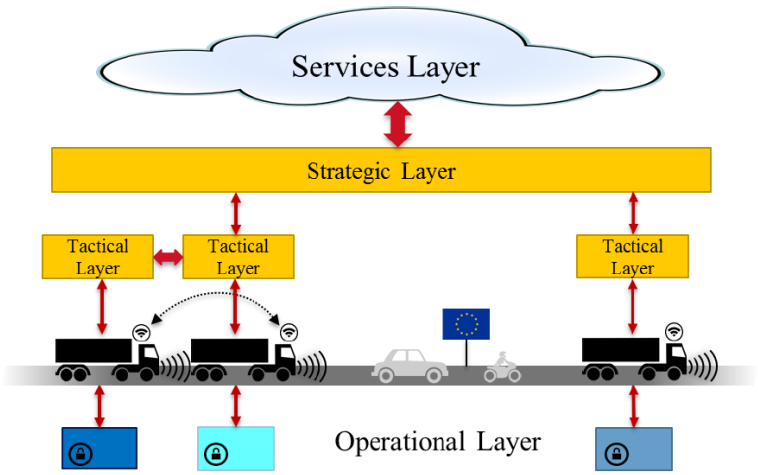


Figure 2.5: Layered Truck Platooning - ENSEMBLE [4]

The functionality of each of the layer is explained as follows:

**Operational Layer:** The Operational layer handles the activities of each of the individual vehicle in the platoon. It is specific to each vehicle and it deals with the vehicle activities such as steering, braking, manoeuvring etc. This layer is responsible for the control of the vehicle such that

it remains in the platoon i.e it maintains the appropriate inter-vehicle distance and follows the platoon behaviour for the entire platoon distance.

**Tactical Layer:** The tactical layer is responsible for the platoon forming and the dissolution. When a new truck wants to join a platoon or leave a platoon or dissolve the entire platoon, this layer coordinates these activities. This layer utilises and handles the Vehicle-to-vehicle communication through which the platoon forming and dissolution is facilitated. Moreover, this layer takes care of the cohesion of the platoon during strenuous road conditions.

**Strategic Layer:** The strategic layer is responsible for the decision-making in the platooning activities such as forming, dissolving, determining the platoon velocity, deciding the inter-vehicle distances etc. The layer is centralised such that it can communicate with all the vehicles and also called as traffic control centres. It also focuses on the optimisation based on the travel times, fuel consumption etc.

**Service Layer:** The Service layer focuses on the end-user side. This handles the new initiatives and the logistical operation demands. Hence, not much functionality is handled by this layer.

These four layer coordinate together enabling Truck platooning. The software for the Truck platooning is really complex and hence it is useful to know the underlying architecture behind it. Alone the software is not enough for the HIL simulation based testing. In any method of verification test cases are necessary. They are the basic building block of any testing. There are several methods to formulate test cases but the one which is widely used in the recent testing of autonomous automotive systems is the Scenario based testing and which is mainly based on the Use Cases.

## 2.3 Scenario based testing and Use Cases

Scenario based testing is a testing methodology which is mainly used in the system testing. This scenario based testing is particularly used in the development and also mainly in verification and validation of the automotive systems. The scenarios can also be captured in the form of Use Cases. Hence, the Use Cases should be determined upfront because this is used also in the development phase. The Use Cases form an essential input for the generation or creation of the test cases and then used in simulation based testing methodologies. The scenario based testing works such that scenarios are specified in natural language and then Use Cases are derived from then which is a sequence of interactions and then formal test case is derived from the Use Case for the testing purpose. This thesis focuses on the Scenario based testing in HIL. A number of definitions for scenarios and Use Cases are available in the literature. Scenarios are defined in [23] as :

*"An ordered set of interactions between partners, usually between a system and a set of actors external to the system. May comprise a concrete sequence of interaction steps (instance scenario) or a set of possible interaction steps (type scenario)."*

A Use Case is defined in [24] as:

*"A sequence of interactions between an actor (or actors) and a system triggered by a specific actor, which produces a result for an actor."*

Formal test cases can be derived from the Use Case description or in few instances from the scenario description for the testing purposes. Hence, Use Cases form an important input for the HIL testing. In this dissertation one of the Use Cases for the Truck platooning will be considered for the evaluation of the simulators. This Use Case will be first converted into a formal test case which is given as input for the testing. It is important that the Use Case description details all

the expected outputs and also the conditions for the executions. It should also provide the initial conditions from which the scenario should start. There are plenty of softwares which facilitate converting the Use Case description into formal test cases although the specification should be written by the user. In the HIL setup of TNO, which will be explained in detail in the next subsection, OpenSCENARIO [5] file format will be used to specify the dynamic contents such as speed, conditions etc from the Use Case descriptions to convert them into a formal test case.

### 2.3.1 OpenSCENARIO and OpenDRIVE

OpenSCENARIO and OpenDRIVE, which are open file formats which are written in XML, are being increasingly utilised in the scenario based testing of the automotive systems. These format specifiers enable the functional scenarios described in the natural language into logical scenarios which are used as test cases in the simulation based testing. OpenSCENARIO is mainly used for specifying the dynamic parameters such as the environmental details, the vehicle parameters and weather conditions. OpenDRIVE is mainly used to specify the road network i.e it helps providing the description of the type of road that will be used in simulation such as straight roads, curved roads, surrounding traffic conditions etc [5]. A diagrammatic representation of the specification provided in [5] is depicted in Figure 2.6:

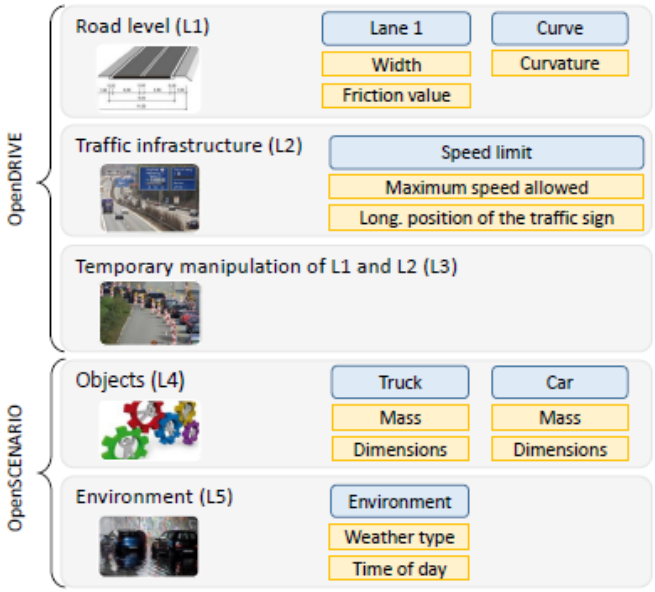


Figure 2.6: OpenSCENARIO and OpenDRIVE [5]

As observed in the Figure 2.6 the entire format specification is represented in a five layer model. The first three layer involves the OpenDRIVE specification. As explained above, the layer 1 is used to specify the road specifications such as the road length, the width, the curvature if a curved road or even a tunnel. The second layer also falls under the OpenDRIVE specification which deals with the details of the traffic infrastructure such as the speed limits, placement of the traffic signs etc. The third layer is for the temporary modifications of the specifications in the L1 and L2 for a particular test scenario. This constitutes a separate file with the road specifications and with the extension .xodr. The format specification for both the OpenSCENARIO and OpenDRIVE are in XML format.

The remaining two layers are for specifying the dynamic contents which includes the vehicle

and environment specifications. The L4 is used for defining the vehicle characteristics such as the dimensions, speed and also used to specify some conditions. The L5 is used to provide the environmental conditions such as the Weather and time details etc. The OpenSCENARIO files are stored with the extension .xosc. These two files thus enable to specify the scenario for the testing which is a test case.

## 2.4 Simulations and Simulators

As described in Section 2.1 Simulators play a key role in the HIL testing. Simulations create a virtual environment close to reality. Simulators are basically the agents that provide simulation. In [25] Simulation is defined as *the approximate imitation of the operation of a system over time*. Since it is an approximation of the actual behaviour, simulation can be used to do performance analysis, find optimization parameters and observe the actual behaviour and make changes. Moreover, by varying the simulation parameters several predictions can be made about the behaviour of the system under study. Hence, this is a really useful process, especially if the system is complex, safety-critical and difficult to manufacture. Since simulations show behaviour of a process over time, several design decisions can also be drawn from simulations.

The validity of the design decisions can be also verified with simulations. Recently 3-D simulators are developed which helps in better understanding of the design and analysis of a process or a product [26, 27]. These powerful functionalities that a simulators have make them an indispensable tool in the Model-based software development. As explained in Section 2.1 and in Figure 2.2 simulation is used in almost every stages of verification. Thus the HIL testing is no exception. The simulators help visualise the environment of autonomous vehicle in the HIL testing. Thus any misbehaviour of the system or crashes can be easily found out based on the visualisations. However, not all simulators can satisfy the underlying system requirements. For example, testing of airplane system has a completely different set of simulator requirements, or HIL testing of a nuclear reactor can have a completely different set of simulator requirements. Hence the simulator must be chosen based on the application. Thus choice of the simulator also play a major role. Given that the system under consideration for HIL testing is an Autonomous vehicle, the simulators which are capable of simulating autonomous vehicles and the environment around them is necessary.

There are several simulators which are available for simulating autonomous vehicles. In this thesis three of the major simulators in the autonomous vehicles are considered for the evaluation. These simulators are compatible with the HIL setup at the TNO and also compatible with the ROS middleware. A brief description of the simulators and their underlying simulation engine are described below:

### 2.4.1 CARLA Simulator

CARLA (CAR Learning to Act) [28] is an open-source simulator which is used mainly in the simulations of autonomous vehicles and urban driving. CARLA simulator is built on top of the Unreal Engine 4 simulation Engine. The simulation engine itself is also open-source and the usage of physics simulation engine enables CARLA to create new vehicles and the surrounding environment. CARLA basically is a server-client system in which the server runs the simulation and also responsible for the rendering the environment. The client is made of a python API and this acts as a interface where commands and meta-commands are sent to the server and the data such as sensor data is returned. The client API uses sockets to communicate to the server. The commands that are sent to the server could be the steering commands, braking commands etc. The meta-commands are for controlling the simulation settings and the environmental settings. These commands are passed to the server via the sockets and the server renders the scene and runs the

simulation accordingly.

CARLA focuses on the development, performance analysis and training of the autonomous driving systems [28]. These features make this simulator a major choice for simulations that are required for using Artificial Intelligence(AI) in automotive systems. CARLA provides the exact locations and bounding boxes for all the dynamic objects in the simulation. This make it really suitable for the training and evaluation of the driving policies. Additionally, CARLA provides two urban environments with which the Machine learning models can be trained and tested:

i) Town 1 with a distance of 2.9 km of drivable roads. This is used for the training.
ii) Town 2 with a distance of 1.4 km of drivable roads. This is used for testing.

This simulator is considered for the analysis owing to its increasing support for OpenDRIVE and OpenSCENARIO format and is also compatible with the TNO HIL setup.

### 2.4.2 LGSVL Simulator

LGSVL simulator [15] is a simulation tool which is mainly used to facilitate the development and testing of autonomous driving systems. Like CARLA, LGSVL is also built on top of a simulation/game engine. The underlying simulation engine is the Unity game engine. Unity is a popular simulator choice for the creation of digital twins. Hence, this makes this simulator also a good choice for the digital twin creation. The underlying architecture in the LGSVL simulator has a ROS bridge which connects the Sensor and environmental parameters and the simulation features with the algorithms for autonomous driving such as the Lane detection, object detection etc. The simulator also supports the latest ROS2 bridge.

The LGSVL simulator provides a python API which is used to control the dynamic objects. Moreover, the environmental features such as the weather conditions, the time of day can be dynamically changed with the help of the python API. Moreover, the LGSVL simulator is capable of data generation for machine learning purposes. The simulator utilises the python API for data collection and thus provides the data collection during the simulation.

The features mentioned above makes this simulator a suitable choice for the autonomous vehicle systems and thus for the evaluation of the simulator for the truck platooning application in this thesis.

### 2.4.3 GAZEBO Simulator

GAZEBO is a robotic simulator and it is one of the most widely used simulator for the robotic simulations. Recently, this became a popular simulator in simulation of autonomous vehicles because of its widespread ROS support and also because it simulates robots which by themselves are autonomous agents.

The aim of GAZEBO simulator as described in [6] is to facilitate addition of new sensors, new agents, actuators etc. Thus it enables creation of the world around the robot as realistic as possible which in-turn helps in the development and testing of the robots. This principle can also be applied to autonomous vehicles. Hence the architecture of the GAZEBO has an API which makes it easier for users to create new sensors as a plugin. The layer below this API is the layer which handles the physics and visualization of the simulation. The physics engine on which the simulator was built is the Open Dynamics Engine (ODE). This engine focuses on the simulation of the dynamics and kinematics of the rigid bodies. GAZEBO has a thin layer of abstraction between the ODE engine and the GAZEBO models. This enables creation of complex planes while maintaining the functionality provided by the underlying physics engine. The architecture of Gazebo is represented as in Figure 2.7:

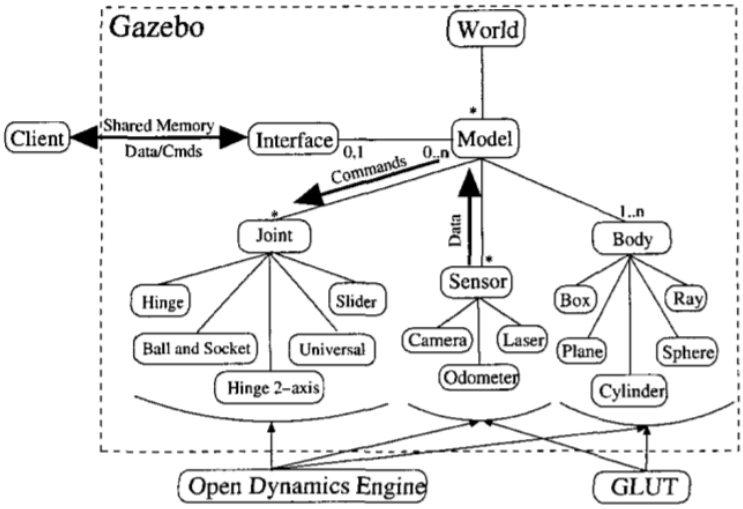


Figure 2.7: GAZEBO Architecture [6]

As can be observed the world is the simulation environment which has one or more models which can make use of the features of the ODE engine and at the same time create complex planes using the client API. GLUT is nothing but a toolkit used to create interactive 2D and 3D applications. Gazebo is also based on the server-client system like CARLA. The main advantage of the GAZEBO is its support for the ROS. It is one of the widely used simulators with ROS middleware. The Gazebo provides a package called GAZEBO ROS package which provides interface between Gazebo and ROS framework. This is assisted by the use of Topics and plugins. Topics help run the server remotely and the plugins help in providing access to the API to specify the parameters.

Gazebo thus has significant ROS middleware support because it has a dedicated package to support ROS framework and has provisions to add any additional sensors, actuators by means of plugins. Moreover, it is possible to make the sensors realistic by adding noise. These all things make Gazebo a suitable simulator for the evaluation and its suitability for Truck platooning.

## 2.5 TNO HIL setup

TNO organisation has a Hardware-in-Loop test bed where they develop and test truck platooning applications. Currently, there are 3 vehicles in a virtual environment that responds to the control inputs given by the driver or hardware. Thus these vehicles behave accordingly as they are actually behaving in the real world. This behaviour is visualised in the simulation and thus it is used for the development and testing of the complex autonomous automotive systems. The Integrated vehicle Safety (IVS) department of TNO develops control algorithms in MATLAB/Simulink and these are tested in the virtual environment using the HIL setup.

There are three ways that a user can manually provide the control input. One is through the steering wheel, through pedals and through the Human Machine Interface (HMI). As can be seen from the Figure 2.8 the input can be manually provided by means of the three interfaces and the corresponding behaviour can be visualised using the simulator. These control inputs can also be specified in terms of software. This is where the scenario based testing, which is the scope of this thesis, occurs. The scenario files provide the control inputs and the user can observe the behaviour

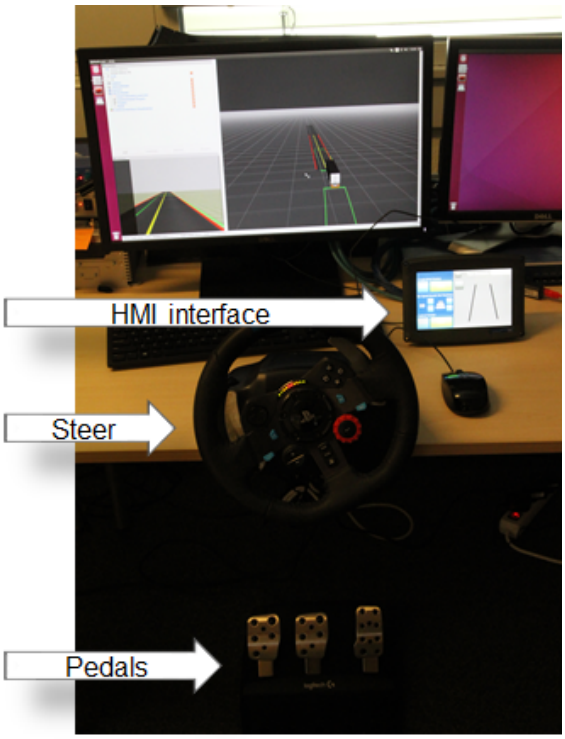


Figure 2.8: TNO test bed setup and visualisation

of the vehicle through the simulator.

The architecture of the TNO HIL test bed is figurated in the Figure 2.9. The computers which emulate the trucks are connected by the ethernet and the inter-process communication is facilitated by the ROS. The architecture has one computer per vehicle to run the platooning application and one computer to run the simulation environment. Moreover, it has the hardware frames that facilitate the vehicle-to-vehicle communication, HMIs etc. These are connected as plug and play devices.

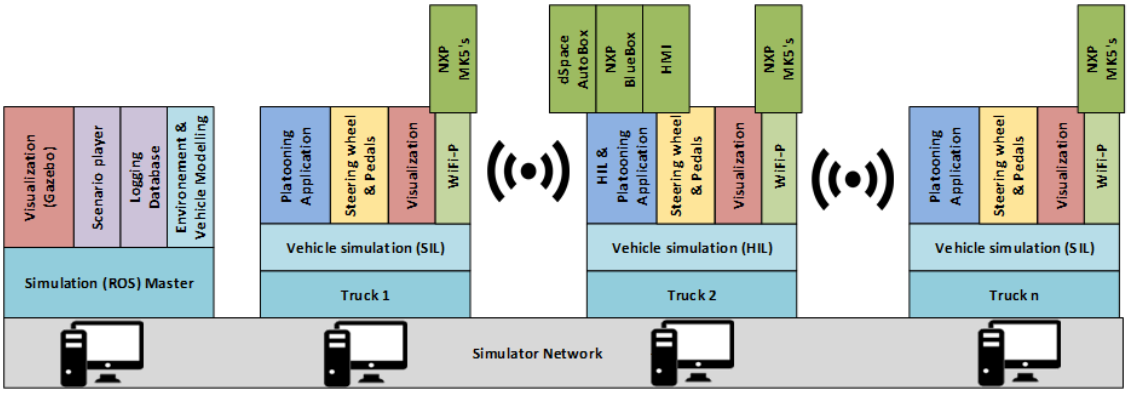


Figure 2.9: TNO HIL Architecture

As observed from the Figure 2.9 there are 3 trucks currently in the TNO HIL testbed. Each computer emulates the behaviour of the truck. Through the Wi-fi connection the vehicle-to-vehicle communication takes place which is an integral part of the truck platooning application. The scope

of this thesis lies in the visualisation area. This is where the simulators come into play. The simulators emulate the behaviour of the trucks which will be their actual behaviour on road. As seen from the figure, ROS [29] acts as a middleware which helps in the inter-process communication and it is based on publish/subscribe mechanism through which data can be sent and received. The simulator subscribes to the ROS topics and also sometimes publishes topic if data is required from the simulator. Thus ROS is an integral part of the TNO HIL testbed.

Knowing the HIL setup of the TNO, it is important that the choice of simulator should be such that it helps visualising the actual behaviour of the vehicle and at the same time has compatibility with ROS environment.

## 2.6 Related Work

Truck platooning is a relatively new concept which has not yet been practically put to use because of several safety restrictions. Hence, there are relevant works towards different methods of truck platooning implementation and verification. On the other hand, simulators for autonomous vehicles is also a widely researched topic and there are several literature works with respect to this field.

In [30] the autonomous vehicle platooning is verified formally. In this research paper, a model of the truck platooning was created and using process automata theory the safety requirements are verified. However, only the Use Cases of joining a platoon and leaving a platoon only is considered. The requirements are defined for the Use Cases and the code is generated based on the process automata. In addition to that few other safety requirements are verified. In other formal verification of truck platooning conducted in [31] safety requirements are more focused. The safety requirements are first defined and the Failure Mode and effect analysis is done. Then the process specification is done in mcRL2 software and is verified. This is similar to [30] except that it focuses more on the safety aspects. The outcome of these two works ensure that truck platooning application is completely feasible with some level of automation and not fully automated. However, there is not much work in literature with respect to the entire truck platooning application verification with HIL and especially focusing on the simulators.

On the other hand, several research papers are published which are focused on the aspects of robotic simulators. In [32] a comparison between MORSE and GAZEBO is done for the multi-robot systems. This paper primarily focused on three aspects real-time factor, CPU and GPU load. The capabilities and features of each of these simulators are first indicated and the comparisons are done. As an outcome it was found that MORSE performed better than GAZEBO but it has to be noted that there are several important parameters that have to be considered and not only on the three aspects mentioned above. In [33] another comparison of simulators for the autonomous vehicles are done. However, no experimental verification or comparison is done. The comparisons are done mostly based on the features that are available in each of the autonomous vehicle simulators CARLA, AIRSIM and GTA V.

[34] is focused on the integration of a simulator with ROS environment. In [34] a simulation environment was created with ROS and Gazebo for the simulation of multi-robots and to facilitate the testing. The output of this research is a reliable environment for mobile robots testing and simulation and also the generation of 3D maps. The methodology to integrate Gazebo with ROS in this thesis is a bit similar to this paper. In [35] the approach to simulate autonomous vehicles in urban traffic scenarios is considered. In this the features that are required for the simulators are mentioned first and a number of simulators are annotated with the presence or absence of the required features. This is one of the very important research paper because they define the characteristics the simulator should have and these characteristics are important for the determination of KPI in this thesis.

It can be observed that though there are several works done related to the verification of Truck platooning and with the autonomous vehicle simulation none focuses extensively on the effectiveness of a simulator in HIL and for a truck platooning application. This is the major goal of this dissertation which focuses on doing a comprehensive analysis and finding the effective simulator for HIL based verification of Truck platooning application.

With the background of HIL testing and the truck platooning application, the importance of the performance Indicators of the simulators and the selection of Use Case for the evaluation can be understood. The next chapters explain the determination of the Key Performance Indicators, their weights and the implementation of the selected scenario in the three simulators.

# Chapter 3

# Key Performance Indicators and Weights Determination

The first research question of this dissertation focuses on the requirements or the Key-Performance Indicators of the simulator. The KPI determination plays an important role in the choice of the simulator and also to measure how effective the simulator is for the system under consideration. These KPIs are determined after reviewing the literature, consultation with the domain experts in TNO and also based on the indicators the Truck platooning application demands. The determination of KPI alone is not enough. Depending on the application, cost availability and based on the requirements the KPIs can have different impact. Thus for each KPIs the weights have to be determined. Hence, this chapter details the Key performance indicators and the method to determine the weights.

## 3.1 Key-performance Indicators

In [35] some of the main characteristics that a autonomous vehicle simulator should have, is detailed. In addition to that there are few KPIs that the Truck platooning application demands and there are few KPIs that the HIL setup in TNO demands. The diverse requirements of the KPIs that an application demands is one of the major reasons that the choice of an effective simulator for a complex safety-critical embedded system is not an easy task. The KPIs which are considered for the evaluation are listed below:

### 3.1.1 Fidelity

Fidelity is the parameter which indicates how close the simulation is to the reality. It basically indicates the realism in the simulation. Especially in a safety-critical system, such as truck platooning and autonomous driving systems, high fidelity is a really important parameter. High Fidelity means the simulation is more reliable.

### 3.1.2 Sensor availability and Emulation

In a virtual test bed, the data from the sensors are really important for the analysis and also for the control of the actuators. Therefore, it is necessary that the simulator has to emulate certain sensors which are in the actual vehicle. Some of the sensors include Camera, LIDAR, RADAR etc. Moreover, it is not realistic to expect a simulator to emulate all the required sensors. However, the simulator must have the capability of writing a sensor plugin and thus can be integrated with the simulation.

### 3.1.3 Sensor Noise Consideration

Sensors, that are in the actual vehicle have sensor noises. For example, an image from camera contains various noises [36] and almost all vehicles have an optical camera sensor. Thus realistically any data from the sensor is prone to noise. Thus, a simulator providing sensor is not completely realistic unless it also can take sensor noises into consideration.

### 3.1.4 V2X communication utilities

This is a KPI which is really important for the Truck platooning application. The V2X communication is a generic term for Vehicle-to-Vehicle Communication, Vehicle-to-infrastructure communication etc. Truck platooning relies on the effective communication between the trucks in platoon communicating the status and data between them the entire duration on the road. Sometimes, the trucks also has to transfer data to some data center. Hence the simulator should be capable of simulating the V2X communications.

### 3.1.5 Real-time factor

In a few instances simulators can run actually slower than the real-time execution. Hence, usually there are two timelines in a simulation. One is the simulation time and the real time. Real-time factor indicates at which pace the simulation is running with respect to real-time[32]. If the factor is 1, then the simulator runs at same pace as in real-time. It is given by the following expression:

$$Realtimefactor = \frac{Simulationtime}{Realtime}$$

### 3.1.6 Collision Detection

Collision detection is a KPI that falls under the safety aspect. Since the required simulator deals with trucks and other vehicles in the traffic some anomalies could lead to collision. Hence, this is an important feature that almost any automotive vehicle simulator should have.

### 3.1.7 3D simulation/Visualisation and Resource Usage

3D simulation or visualisation is one of the basic feature that should be imperative in almost all the automotive systems simulator. Though this is not a application specific KPI, it is a basic requirement that all the simulator must have. Moreover, Simulators usually have a heavy load on the machine in which it is being simulated, especially 3D simulation. In addition to the CPU consumption, because of the higher graphics, simulators also have high GPU usage as well. A simulator which really consumes a lot of load is not completely ideal. Hence, this factor should also be considered.

### 3.1.8 Large-Scale Traffic Generation

In real world scenario, there will be several other vehicles on the road where the trucks drive in a platoon. Therefore, there are several instances where another vehicle which is not in the platoon has to cut-in and the platoon has to make necessary modifications. Hence, it is important that the simulator is capable of generating and handling large-scale traffic.

### 3.1.9 Multi-agent simulation

Multi-agent simulation is where the simulator can create and control more than one agent. In the case of truck platooning more than one truck which is essential for the system under consideration. But as the number of agent increases there will be other impacts such as the system load and real

time factor will be decreased. Therefore, an optimal trade-off should be made. It is to be noted that it is not in the scope of this thesis.

### 3.1.10 Failure/Fault Simulation

This is another KPI related to the safety aspect. Majority of the safety-critical systems are designed as fault-tolerant systems. Therefore, the simulator should be able to simulate the fault tolerance i.e. it should be be able to induce failure such as corrupt the data from the sensor to the actuator and thus helping observe the behaviour.

### 3.1.11 Vehicle model and Environmental model creation features

Autonomous vehicle simulators by default have few vehicle models predefined. Similarly, they also have few environmental models where a particular setup of roadways is built-in. For instance, as mentioned in Section 2.4.1 CARLA has two urban environments by default for training and testing. It is to be noted that not all vehicle models and environmental models are available in simulators in default. Thus the requirement of the simulator is to facilitate or provide API features to create the user's own vehicle and environmental models.

### 3.1.12 Environment affecting sensors and physics of the vehicle

In reality both the sensors and vehicles are affected by the harsh weather conditions. Many simulators do not necessarily take these factors into account while simulating. But these factors are important to achieve a high fidelity simulation. For instance a harsh weather conditions such as foggy weather or darkness affect the camera visibility and intense weather causes echoes in laser scanners[35]. Moreover, weather and road conditions affect the physics of the vehicle. Snowy and rainy conditions make the road more slippery. The simulator should consider these effects while calculating the sensor data. This brings more realism into the simulation.

### 3.1.13 Ease of plugin creation and simulation control

Extension capabilities of a simulator imply creation of additional plugins and libraries if the required aspects are not available. For instance, if a simulator has a RADAR and the user wanted a different modification of the radar, it should be able to modify the radar. Moreover, if there is no Radar in the simulator there should be provision to add a RADAR in the form of a plugin or a library. This is an important KPI to have because a simulator cannot provide everything by default and the user requirements also vary.

It is also important to have ability to control the simulation stepd such as pause the simulation, step forwards to a specific time or run faster than 1:1 time etc. These features helps analysing the performance even better if we have several control abilities of the simulator.

### 3.1.14 OpenSCENARIO and OpenDRIVE support

This KPI is specific to the TNO HIL setup. The two open file formats are useful in specifying the scenario for the scenario based testing. As mentioned in Section 2.3.1 OpenSCENARIO and OpenDRIVE are used to specify the static and dynamic contents of the scenario. This is an useful KPI for projets in which Scenario based testing is used in HIL.

### 3.1.15 Continuous Integration (CI) facilitation

This KPI is another aspect specific to the TNO HIL setup and also in many other industries. Several industries adopt CI as part of their software management process. It is convenient if the simulator can be run in headless mode as part of CI pipeline. Though this is not an necessary KPI, it is a KPI which is nice to have.

## 3.2 Method to assign weights to KPIs

To evaluate the performance of a simulator, first the KPIs are identified. Based on these KPIs the simulators can be benchmarked and also a lack thereof could help improving the simulator. It is to be noted that not all the KPIs play an equally important role. The KPIs vary depending on the application, requirements and also on the user convenience. Therefore, it is important to find how much each KPI weigh inorder to know their relative importance. This way the performance parameters can be efficiently benchmarked.

### 3.2.1 Key Performance Area Segregation

The list of Key performance indicators is quite extensive. Moreover, not all the KPIs have the same weight. Literature suggest categorizing the KPIs into certain key performance areas. For example in [37] the KPIs are categorized into 5 KPAs. Similarly, in [38] the KPIs are categorized into 6 key performance areas. In this dissertation, we will categorize the key performance indicators into 5 key-performance Areas. This categorization of the KPIs to KPAs help in building a hierarchy which is useful in the method used for the determination of the weights. The categorization is classified as below:

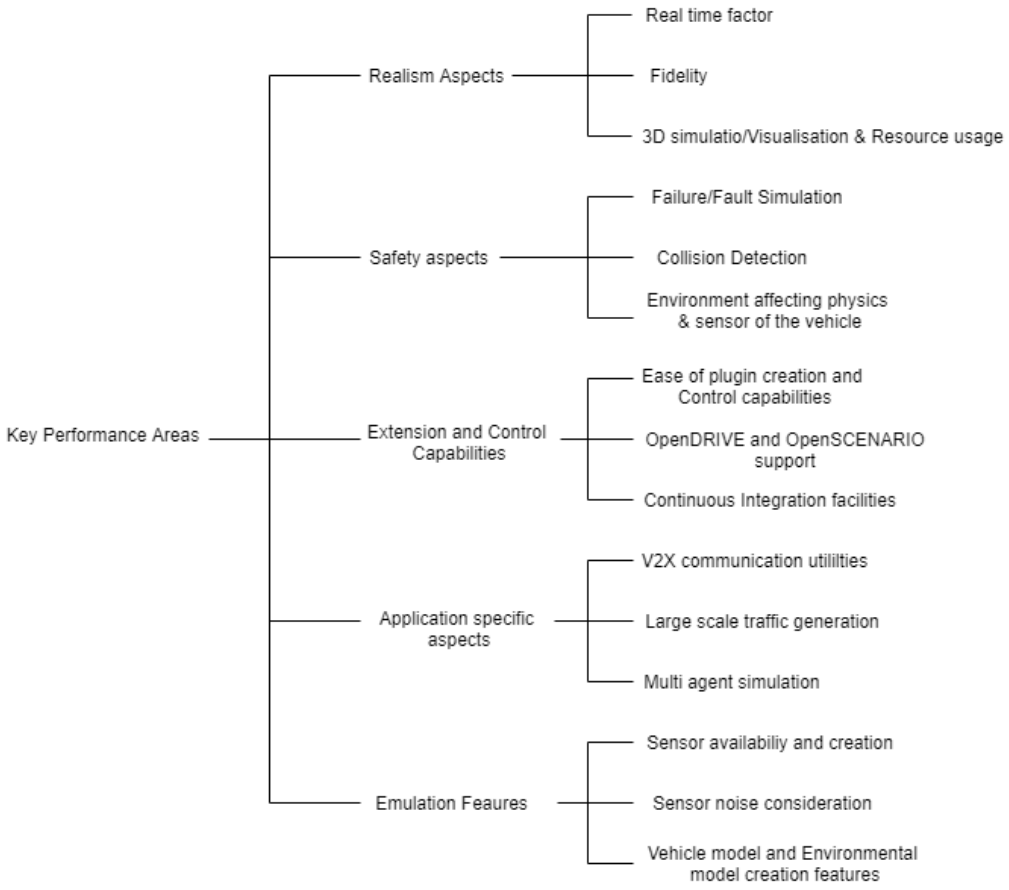


Figure 3.1: KPA segregation and Hierarchy creation

### 3.2.2 Analytical Hierarchy Process

Analytical Hierarchy process is one of the most used methods for decision making in a Multi Criteria Decision Making paradigm[39]. It is an efficient method for modelling and solving complex decision making problems across various domains. In this dissertation we make use of the Analytical Hierarchy Process (AHP) to assign weights to the key performance indicators. There are four major steps involved in the Analytical hierarchy Process [40]:

1) Define the goal of the problem and the required parameters(KPIs in this case)
2) Construct and structure the problem as a hierarchy
3) Perform pairwise comparisons among the categories and at all levels within the hierarchy using the 9 point scale for the process
4) Compute the relative weights of the criteria

At the end of these four steps we will obtain the weights of the key performance Indicators. In the third step of the AHP the criteria have to be rated on a 9 point scale. The 9 point scale is tabulated below:

| **Scale** | **Numerical Rating** |
|---|---|
| **Extremely preferred** | **9** |
| Very strong to extremely preferred | 8 |
| **Very strongly preferred** | **7** |
| Strong to very strongly preferred | 6 |
| **Strongly preferred** | **5** |
| Moderate to Strongly preferred | 4 |
| **Moderately preferred** | **3** |
| Equally to Moderately preferred | 2 |
| **Equally Preferred** | **1** |

Table 3.1: Saaty's Relative importance Scale

It is important to note that the use of odd numbers in the Saaty's scale is preferred. Because, this enables clear distinction between the preference of one criterion over the other.

This pairwise comparison matrix has to be filled with the relative importance based on the 9 point scale listed above. An example of the pairwise comparison matrix filling is illustrated below:

| | KPI/KPA 1 | KPI/KPA 2 |
|---|---|---|
| KPI/KPA 1 | 1 | Numerical Rating |
| KPI/KPA 2 | 1/Numerical Rating | 1 |

Table 3.2: Pairwise comparison Matrix

Now that the 9 point scale is explained, it is important to form the hierarchy based on the the major goal. The AHP process works first by formulating the goal and then the criteria are identified. Additionally, more levels of hierarchy can be found out and alternative criterion for every criterion can be identified. But the scope of this thesis requires only two levels of hierarchy which is observed in Figure 3.1. Generally, the Analytical Hierarchy process has the following structure:

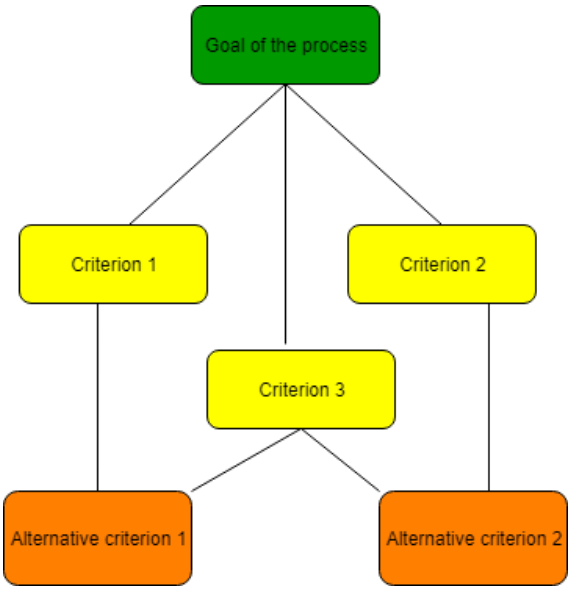


Figure 3.2: Analytical Hierarchy Process - Hierarchy structure

In this dissertation a hierarchy like structure in Figure 3.2 is already constructed in the Figure 3.1. The goal of the process in this case is Finding an effective simulator. The first layer of hierarchy has the Key performance areas and the second layer of hierarchy has the Key performance Indicators. Therefore, the pairwise comparisons of the Key performance areas will be done and also the pairwise comparison within each of the Key performance areas i.e between the KPIs within each hierarchy will be done.

### 3.2.3 KPI weight Calculation

Once the hierarchy is determined, the pairwise comparison matrix need to be filled out. For this purpose, a survey is conducted with the domain experts in TNO. A total of 6 domain experts who work in different platforms of truck platooning such as development, simulation testing, integration are considered for the survey. The KPIs are explained and the details of the pairwise matrix were provided. After the survey, the pairwise matrix is obtained from all the domain experts. The average numerical rating value is considered for the final calculation. After obtaining the pairwise comparison matrix, the *Eigen vector* of the matrix is found. The first step is to normalize the comparison matrix. This is done by dividing the each value in the matrix by the sum of its column values. The next step is to calculate the Eigen vector. This gives the contribution or the weight of each of the criterion to the overall goal. The Eigen vector gives the relative weights between the KPIs or KPAs in each layer. Thus the final Eigen vector gives the weights of each KPAs and each KPIs inside the KPAs.

One thing that has to be taken into consideration is the data inconsistencies. This is because a diverse group of people take part in the survey and this consistency check is to ensure whether the decision makers are consistent in their choices. Hence, in this process we need to calculate the consistency index. The consistency index is dependent on the maximum eigen value. The maximum Eigen value is calculated by first multiplying each element of the eigen vector with the respective column total of the original comparison matrix. Finally, the result of each of the multiplication is summed together thus resulting in the maximum Eigen value.

The consistency index is then calculated using the formula[41]:

$$CI = \frac{\lambda_{max} - n}{n - 1}$$

where $\lambda_{max}$ is the maximum eigen value and $n$ is the number of criteria that is used. This way the consistency Index is found. [41] also defines Consistency Rate which determines whether the data collected is consistent or not. It is given by the ratio of Consistency Index (CI) and a term Random consistency Index (RI). The RI values are fixed for each number of criteria. The RI values for various number of criteria are tabulated below:

| **n** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **RI** | 0 | 0 | 0.58 | 0.9 | 1.12 | 1.24 | 1.32 | 1.41 | 1.45 | 1.49 |

Table 3.3: Random Consistency Index (RI) values

The pairwise comparison matrix is considered consistent if the ratio between CI and RI is lesser than 10 percent.

$$CR = \frac{CI}{RI} < 10\%$$

If the matrix is consistent and if the Eigen vectors are calculated then we obtain the weights. If we want to obtain the overall weights of each of the KPIs then we have to multiply the weight of the KPI obtained from the pairwise comparison matrices with the corresponding weight of the KPA. In this way the weights of the KPI are found out.

### 3.2.4 Experimental Results - Weight Calculation

This subsection describes in detail about the weight of the KPIs that are found as a result of calculation based on the above mentioned methods. Moreover, it also gives the final weight of each of the KPIs.

| **KPA Matrix** | Realism Aspects | Safety Aspects | Extension and Control Capabilities | Application Specific aspects | Emulation features | **KPI** |
|---|---|---|---|---|---|---|
| Realism Aspects | 1 | 2.131933 | 2.223833 | 1.991 | 1.220833 | **0.179216** |
| Safety Aspects | 2.471117 | 1 | 3.3055 | 2.005 | 2.543333 | **0.227063** |
| Extension and Control Capabilities | 2.329333 | 1.0875 | 1 | 2.943333 | 3.088333 | **0.206891** |
| Application Specific aspects | 3.398833 | 3.843333 | 1.044667 | 1 | 3.388333 | **0.246577** |
| Emulation features | 2.708333 | 1.159117 | 1.5015 | 0.821 | 1 | **0.140252** |

Table 3.4: Key Performance Areas Pairwise Comparison Matrix

The following tables below lists the pairwise comparison matrix of each of the Key performance Indicators in the Key performance Areas.

| **Realism Aspects** | Real-time factor | Fidelity | 3D simulation/ Visualisation & Resource Usage | **KPI** |
|---|---|---|---|---|
| Real-time factor | 1 | 0.206833 | 1.626333 | **0.141354** |
| Fidelity | 5.6666667 | 1 | 4.1905 | **0.533797** |
| 3D simulation/ Visualisation & Resource Usage | 2.5888333 | 1.464 | 1 | **0.324849** |

Table 3.5: Realism Aspects KPI pairwise comparison matrix

| **Safety Aspects** | Failure&Fault simulation | Collision Detection | Environment affecting sensor& physics of the vehicle | **KPI** |
|---|---|---|---|---|
| Failure&Fault simulation | 1 | 3.054167 | 1.301 | **0.314188** |
| Collision Detection | 2.3285 | 1 | 1.13995 | **0.285828** |
| Environment affecting sensor&physics of the vehicle | 2.533333 | 3.7 | 1 | **0.399985** |

Table 3.6: Safety Aspects pairwise comparison matrix

| **Extension and Control Capabilities** | Ease of plugin creation and Control capabilities | OpenScenario & OpenDrive support | Continuous Integration facilities(CI) | **KPI** |
|---|---|---|---|---|
| Ease of plugin creation and Control capabilities | 1 | 3.716667 | 3.275833 | **0.432051** |
| OpenScenario & OpenDrive support | 0.8223333 | 1 | 1.743 | **0.20432** |
| Continuous Integration facilities(CI) | 2.7476667 | 2.2555 | 1 | **0.363629** |

Table 3.7: Extension and Control Capabilities pairwise comparison matrix

| **Application specific Aspects** | V2X communication utilities | Large scale traffic generation | Multiagent simulation | **KPI** |
|---|---|---|---|---|
| V2X communication utilities | 1 | 2.039333 | 0.350333 | **0.17199** |
| Large scale traffic generation | 3.143833 | 1 | 1.512167 | **0.338948** |
| Multi agent simulation | 4.833333 | 4.1875 | 1 | **0.489062** |

Table 3.8: Application Specific Aspects Pairwise comparison matrix

| **Emulation Features** | Sensor availability& creation | Sensor noise consideration | Vehicle model and environmental model creation | **KPI** |
|---|---|---|---|---|
| Sensor availability& creation | 1 | 5.833333 | 1.075833 | **0.384994** |
| Sensor noise consideration | 0.1906667 | 1 | 2.131833 | **0.222065** |
| Vehicle model and environmental model creation | 2.3888333 | 2.576333 | 1 | **0.392941** |

Table 3.9: Emulation Features pairwise comparison matrix

## Final KPI weights

As observed from the experimental results in Subsection 3.2.4 the weights of the Key Performance Areas and the weights of the Key performance Indicators within each Key Performance Area is calculated. The final overall weight of each of the KPI is obtained by multiplying the weight of each of the KPI and the weight of its corresponding KPA. The table below gives the final weights of each of the KPIs:

| **Key Performance Indicator** | **Weight** |
|---|---|
| Multi agent simulation | 0.120592 |
| Fidelity | 0.095665 |
| Environment affecting sensor &physics of the vehicle | 0.090822 |
| Ease of plugin creation and Control capabilities | 0.089388 |
| Large scale traffic generation | 0.083577 |
| Continuous Integration facilities(CI) | 0.075232 |
| Failure&Fault simulation | 0.07134 |
| Collision Detection | 0.064901 |
| 3D simulation/Visualisation & Resource Usage | 0.058218 |
| Vehicle model and environmental model creation | 0.055111 |
| Sensor availability& creation | 0.053996 |
| V2X communication utilities | 0.042409 |
| OpenScenario & OpenDrive support | 0.042272 |
| Sensor noise consideration | 0.031145 |
| Real-time factor | 0.025333 |

Table 3.10: Final Weights of the Key Performance Indicators

### Consistency Index - Results

The following table provides the Consistency Ratio of the pairwise comparison matrices that are surveyed. This Consistency Ratio indicates whether the domain experts are consistent or diverse in their choices.

| **Pairwise Matrix** | **Consistency Ratio(CR)** |
|---|---|
| Key Performance Area Matrix | 1.080866 |
| Realism Aspects | 1.6796765 |
| Safety Aspects | 2.098619 |
| Extension and control Capabilities | 2.230731 |
| Application Specific aspects | 2.063314 |
| Emulation Features | 1.8284547 |

Table 3.11: Consistency Ratio - Pairwise Comparison Matrices

As observed from the Table 3.11 the Consistency Index is significantly higher than the maximum limit. The experimental results **does not satisfy** the consistency requirments. However, this can be attributed to the choice of the domain experts where they are chosen in such a way that they they belong to different domains within the design and verification of Truck Platooning. This choice is made keeping in mind that there are several aspects to use of the simulator in the HIL and inputs from people who belong to all domains of the HIL verification is required. This ensures the final weight which is obtained is considered taken into account all aspects of the HIL, current setup of TNO and the key performance Indicators that require more emphasis and that is essential for the new simulator.

It can be noted from the Table 3.4 that the KPA Application specific aspects has more weightage than the other KPAs. Therefore, it is decided to place emphasis on the analysis of the simulators based on the Application specific aspects KPIs while also analysing the other KPIs that are necessary for the implementation of the chosen scenario.

Now that the weights are calculated the next step is to evaluate the Key performance Indicators with respect to the selected scenario. The scenario is implemented in the three simulators and the Application specific aspects KPIs are evaluated.

# Chapter 4

# Scenario Implementation

The research output of the first research question provides the Key performance Indicators and its weights. As detailed in Chapter 3 the Application specific aspects KPA has the highest weight and the KPIs of this KPA category will be analysed and evaluated in detail for answering the research question 2. This chapter discusses in details the description of the scenario, various steps of the scenario implementation and the mapping the steps of the scenario to the KPIs, the implementation details of the scenario in three simulators and also showcases the implementation results. This implementation details and analysis in all the three simulators will be helpful in forming a scoring system and validating the simulators resulting in an effective simulator.

## 4.1 Scenario description and modelling

For the purpose of validating the simulators a Use Case is necessary based on which the simulators will be evaluated. The scenario which is considered for the evaluation is the Cutin scenario. This scenario is considered based on the literature review and discussion with domain experts. Cutin scenario is one of the commonly occurring scenarios in the Truck platooning. Cutin scenario is essentially the situation where two or more trucks are already engaged in a platoon i.e they are already in platooning state and follow each other in close distances. A Cutin happens when a non platoon vehicle performs a lane change manoeuvre and come in close proximity with the lead truck. When this happens, the lead vehicle has to adapt its speed otherwise it will collide with the vehicle which performs the manoeuvre. In-addition to the lead truck the following trucks which are in Co-operative Adaptive Cruise Control (CACC) should also be reducing their speed to avoid collisions. Therefore, the implementation of the scenario in the simulators should involve the simulators simulating the vehicles in platoon, a Cutin from a non-platoon vehicle, the deceleration or braking action of the lead truck and also the remaining trucks in the platoon.

Moreover, to make the scenario more realistic, the scenario is modelled to take place in the real world traffic. In this manner the large scale traffic generation capabilities of the simulators are explored. Due to this addition of traffic simulation, the KPIs of Application Specific aspects such as Multi-agent simulation, Vehicle-Vehicle communication and Large scale traffic simulation can all be considered for the evaluation. This makes the Use Case more appropriate for the evaluation of the simulators.

The scenario is illustrated with the help of a diagram in Figure 4.1:

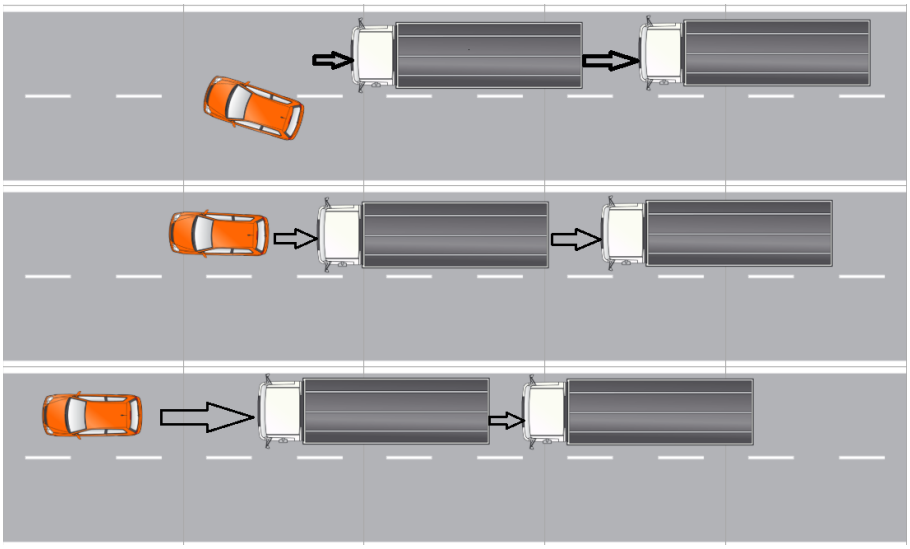

Figure 4.1: Use Case - The Cutin Scenario

From the Figure 4.1, in the first layer, it can be observed that two vehicles are in platoon state which can be inferred from the close distances and in a convoy like movement. For the vehicles to be in a platoon state there must be a predetermined time gap which the vehicles should maintain. Moreover, it should be the case that there should be a greater time gap, which is also predetermined, between the vehicles in platoon and any other participant on the road. Usually, it is the case that there will be a timegap of 1 second for the vehicles in the platoon and a minimum timegap of 2 seconds between platoon and other vehicles in the same lane. A non-platoon vehicle, in this instance a car, tries to cut-in from the second lane. In this case the lead vehicle should decelerate or apply brakes depending on the proximity of the car. This deceleration or braking action should be communicated to all the vehicles in the platoon and thus all the trucks do the action which is done by the lead vehicle and maintain the time gap which is indicated by the diagram in the second layer. The arrows indicate the distance being constantly monitored and communicated to the vehicles. In the third layer, the cut-in scenario took place and the platoon can go back to its normal speed in which they engaged and thus can maintain their previous timegap. This is the description of the scenario which will be used in the evaluation of the simulators.

### 4.1.1 Scenario steps

For the purpose of the implementation the selected scenario is divided into individual scenario steps which makes it easier for implementation and also for analysis. This is used to map the KPIs to the specific steps which will be used for analysis of the simulators and for scoring. The vehicles in the scenario is divided into three. Two vehicles are in the platoon and the third is the target vehicle which performs the lane change manoeuvre and makes the Cutin in front of the lead vehicle in the platoon.

The scenario steps are modelled as illustrated in the Figure 4.2:

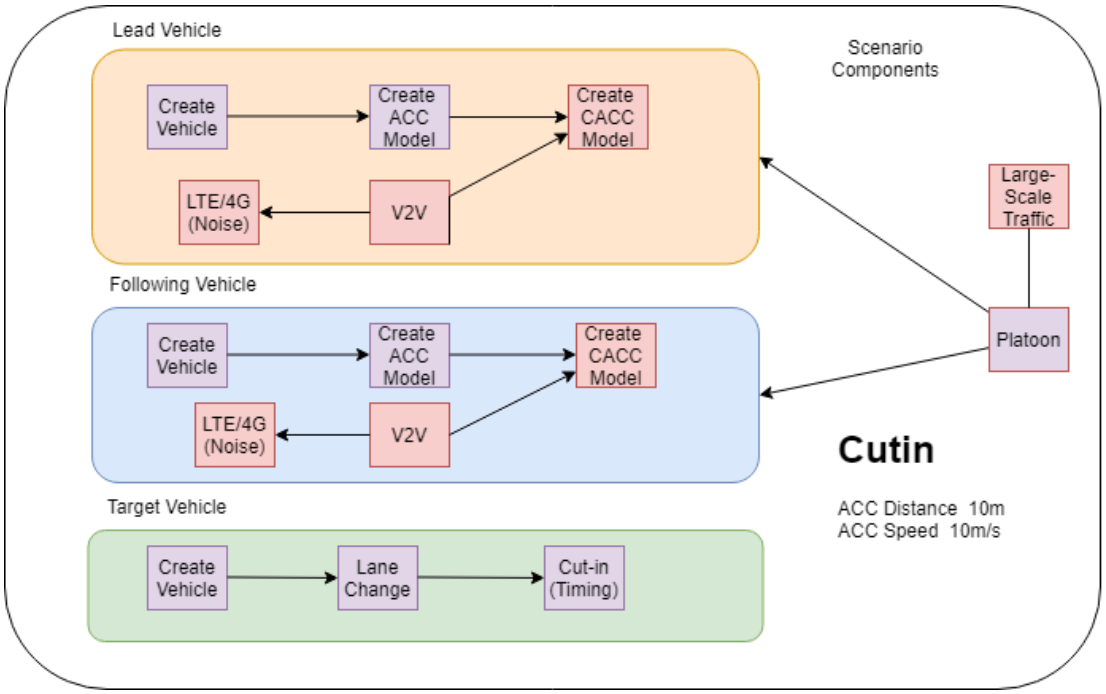


Figure 4.2: Scenario Steps

The division of the Cutin scenario into scenario steps can be observed in the Figure 4.2. Three vehicles are required for the execution of this scenario. *Lead Vehicle* is the lead vehicle in the platoon. The *following vehicle* is the trailing vehicle in the platoon. The *Target vehicle* is the vehicle which performs the cutin. The platoon is a step which is an aggregation of the Lead vehicle and following vehicle. Each vehicle in the platoon has four substeps.

**Create Vehicle:** The *Create Vehicle* step deals with the creation of vehicle in the three simulators. This concerns whether we need to develop a vehicle model or is there already an available vehicle model that we can use. This *Create vehicle* is also concerned about the required sensors that are attached to the vehicle and the sensor creation and emulation properties. Essentially, it deals with creating the environment for the simulation, creating a vehicle, spawning a vehicle at a particular location in the map and attaching sensors to it.

**Create ACC Model:** The *Create ACC Model* deals with the setting of the Adaptive Cruise Control (ACC) mode in each of the vehicles in the platoon. The ACC speed and ACC distance is pre-determined and set which is crucial for the ACC mode. The vehicles initially run at the ACC speed and in-case of a braking action of one vehicle and because of this the other vehicle is within ACC distance then the other vehicle will also perform braking action to maintain the ACC distance.

**Create CACC Model:** Cooperative Adaptive Cruise Control(CACC) is nothing but an extension of the ACC model with Vehicle to Vehicle communication. The location and status of vehicle are constantly communicated to the other vehicles and this each vehicle knows the status of every other vehicles in the platoon. This enables the vehicle to perform appropriate braking, acceleration based on the ACC distance.

**V2V Communication:** The *V2V* step is essential for enabling the Cooperative Adaptive Cruise Control in the platoon. The V2V communication feature should be supported by the simulator and should have the capability to send the status of the vehicle via messages. The communication

could be enabled by means of the LTE/4G. A more efficient V2V simulation model should take into account the noise and loss in signal during communication which occurs in real-time. Hence there is a *LTE/4G(Noise)* block in the V2V scenario step.

**Lane change and Cutin:** The *Lane Change and Cut-in block* scenario steps are specific to the Target vehicle which performs the Cutin. The Lane change is enabled by specifying the timing and the distance it takes for the target vehicle to perform the lane change infront of the lead vehicle in the platoon. Each simulator requires different actions to enable this which will be discussed in detail in further sections.

**Large Scale Traffic:** The selected scenario has the addition of Large Scale Traffic to the Cutin scenario to be more realistic. Hence, the capabilities of the simulators to add large scale traffic will be explored in detail. The simulators will be evaluated on the possibilities and capabilities of handling large scale traffic.

## 4.2 Scenario Implementation

### 4.2.1 Implementation considerations

As mentioned in the previous sections, the selected Use Case is a Cutin scenario in front of the lead vehicle in a platoon with realistic traffic. However, implementing such extensive Use Case in all the three simulators such as Gazebo, CARLA and LGSVL in a short time is not realistic. Therefore, the platoon scenario with Cutin is implemented in this dissertation only with ACC and not with CACC. Moreover, the large scale traffic is also not added to the scenario. However, the V2V communication capabilities and features of the simulator are thoroughly analysed. The possible methods of implementing V2V in these simulators are explored and explained in detail. Similarly, for the large scale traffic generation the capabilities of the simulators are analysed in detail. The possibilities of integration of the chosen simulators with MicroTraffic flow simulators are researched. Therefore, the actual implementation of this dissertation would consist of the implementation of the vehicle platoon with Adaptive Cruise Control and a Cutin happening in front of the the lead vehicle in the platoon. The same implementation is implemented in the three simulators. The CACC, V2V and Large scale traffic simulation capabilities are researched and analysed. Thus, it helps in evaluating the simulators and to provide an appropriate score for the scoring system. As observed in Figure 4.2 the blocks in *blue* are the sections that are implemented and the blocks in *red* are the sections which are researched and analysed. Additionally, the platoon in all the three simulators is implemented with cars. All the simulators have the various car vehicle models by default. Thus, car vehicle model is used. Note that it can be easily extended to the trucks. The implementation of these sections would be a future work.

Other considerations include the setting of the ACC distance and the ACC speed. Similarly, the Cutin is also specified to happen for a specific time instant and a distance. This enables a smooth transition in the cutin. For the current implementation the ACC distance is set as 10m between the vehicles and the ACC speed is set as 10 m/s which is approximately 35 km/hr. Likewise, the Cut in timing is considered as 5.0 seconds and 3.5 meters in distance. This detailing enables performing the lane change manoeuvre over the specified time period and distance. As mentioned in the Figure 4.2 first the vehicles will be created in all the simulators. The vehicles will need to run in a map or an environment. Thus, the environment will be loaded in which the vehicles will be created. Secondly, the necessary sensors need to be attached. This may vary based on the simulator under consideration. It is to be noted that these things vary in all the three simulators which gives an interesting perspective in the analysis. Subsequently, the vehicles are made to run in ACC in which the locations of the each of the vehicle is constantly observed in each of the simulation ticks and if the distance is within the ACC distance corresponding braking action is taken. Thus, a vehicle platoon is created with ACC. The implementation was done incrementally.

First, the ACC model was implemented and verified in the simulation. Then the *Target Vehicle* was added to the ACC implementation and cutin was introduced in the implementation. The results of the implementation is captured in a velocity profile in all the three simulators. The velocity profile will showcase the impact of Cutin and also the ACC implementation which will be discussed in detail in further sections.

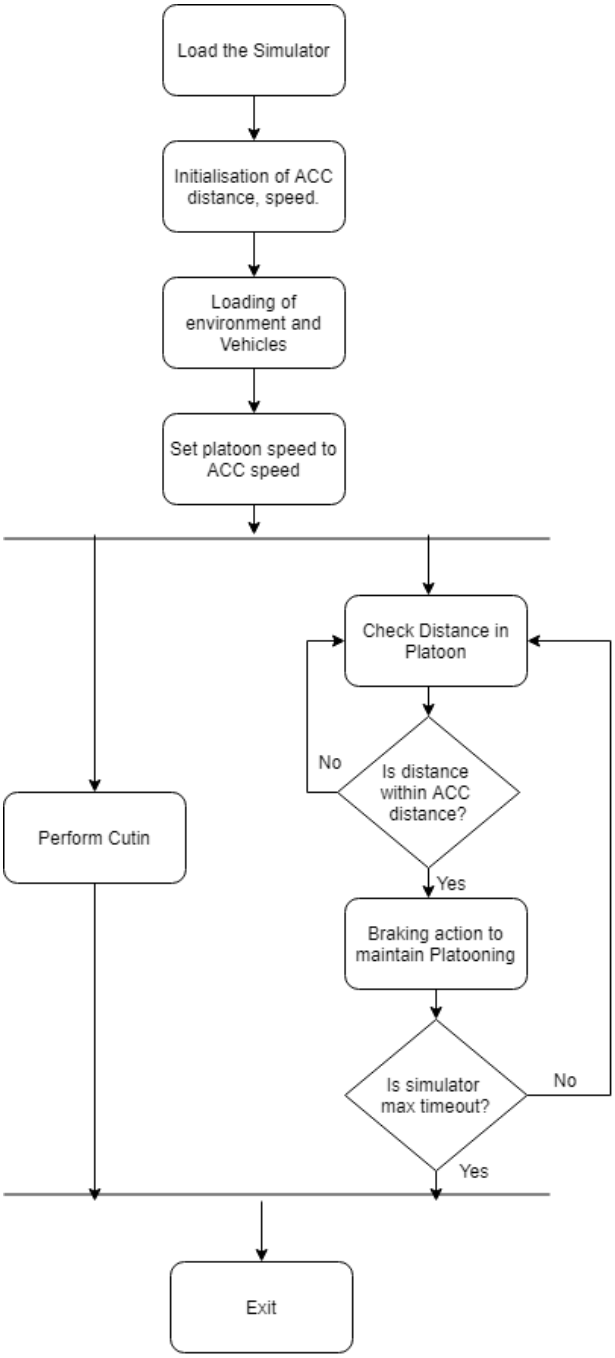


Figure 4.3: Implementation of Cutin Scenario - Flowchart

## 4.3 Cutin Scenario Implementation in CARLA

The flowchart descripted in the Figure 4.3 is the generic flow of the implementation of the cutin scenario. This template is the general flow of code. However, the implementation details vary based on the simulator.

The first step of the implementation is Loading the simulator. CARLA has a python Client API which can be made use of to run the simulations [42]. CARLA simulator has a server-client architecture where the server runs the simulation and the client API can be used to control the simulation. Therefore, to load the simulation a connection needs to be established between the server and the client. This connection is established by means of specifying the IP address which is **"localhost"** by default and the default TCP port which is **2000**. Next step is to load the environment in which the vehicles will be spawned. It is implemented by means of the default maps that are available in CARLA. CARLA simulator by default provides 8 maps[7]. For the current implementation **"Town 01"** is used. It is also possible to create a new map which can be imported using OpenDRIVE. CARLA simulator by default provides several possible vehicle models of which most of them are cars. For the current implementation **Tesla Model3** is used as the vehicle. There is also possibility to randomly choose the vehicles. As mentioned before in the Subsection 4.2.1 the initial ACC speed and distance are set. The schematics of the Town 01 in CARLA simulator and the actual environment is depicted below:

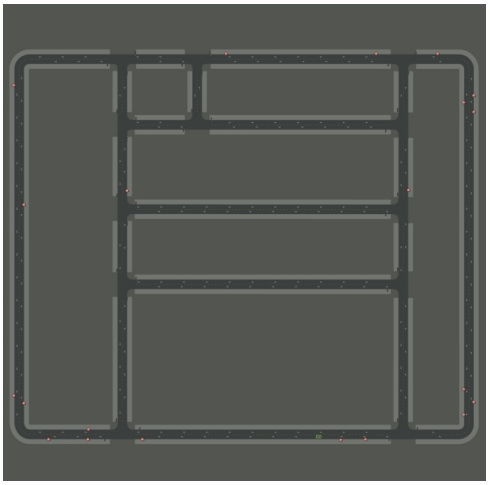

Figure 4.4: Town 01 schematic [7]


Figure 4.5: CARLA environment with loaded vehicles

In addition to the creation of the vehicles various sensors are attached with the vehicles to analyse the sensor emulation capabilities of the CARLA simulator. The sensors such as Camera sensors, Collision detection sensors, radar and lidar sensors are all attached in all the vehicles. These sensors are by default provided by CARLA. But there is also possibility to add user defined sensor through plugins. The initial velocity is set to the ACC speed by means of the *set_velocity* function which is available in the CARLA client API. This sets the initial velocities of the vehicle in the platoon to 10 m/s. The *Target Vehicle* velocity is set to 15 m/s. Subsequently, two threads are created in which one thread performs the cutin using the Vehicle controller in CARLA. CARLA has a *VehicleControl* class which can be used to control the basic vehicle movements such as throttle, braking, steer, reverse and gear shifts. The corresponding class is **carla.VehicleControl**. The respective vehicle is selected and the control parameters are passed as arguments which enables the control of the vehicle. This is made use of to implement the cutin and the braking actions of the vehicles. In the second thread, the distance between the vehicles in the platoon are constantly monitored. Whenever, the distance is within the ACC distance the braking action takes place to maintain the ACC distance. Also, if the cutin happens the distance is also monitored between the *Target vehicle* and the *Lead Vehicle* in the platoon. This enables the scenario to prevent collision with the *Target Vehicle*. Any sort of collision is detected by the collision detection sensor and reported to the terminal.

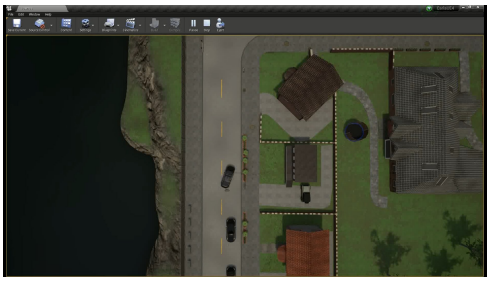

Figure 4.6: Cutin Instant in CARLA

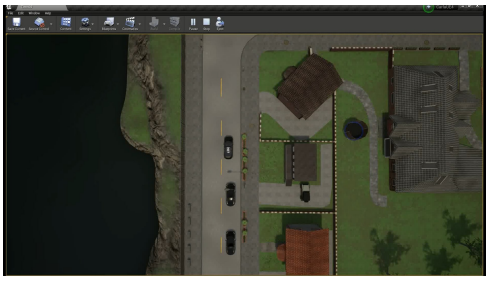

Figure 4.7: Instant after Cutin in CARLA

The Figures 4.6 and 4.7 represent the situation during the cutin and after the cutin in CARLA simulator respectively. It can be seen that the Cutin is performed properly and there is no collision. The cutin behaviour can also be obseved by means of plot of the velocity profile of the vehicles.

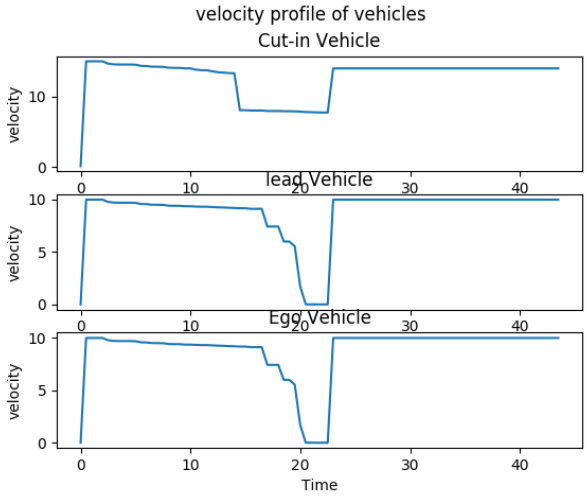


Figure 4.8: Velocity Profile of vehicles - CARLA

The effect of Cutin on the vehicles in the platoon can be clearly observed from the Figure 4.8. The instant of the cutin happening can be observed by the decrease in the speed of the Cutin Vehicle which is the *Target Vehicle*. After the Cutin the lead vehicle decreases its speed to avoid collision and maintain safe distance. Because of the reduction in speed of the Lead vehicle the trailing vehicle which is the ego vehicle will also reduce its speed to maintain ACC distance. Once the Cutin vehicle is safely above the ACC distance the vehicles in the platoon will resume their original platoon speed. Thus the Cutin scenario is successfully implemented in CARLA and the effect is shown by means of the velocity plot.

## 4.4 Cutin Scenario Implementation in LGSVL

The implementation of the Cutin scenario in the LGSVL simulator also follows the flowchart described in the Figure 4.3. Loading the simulator is the first step. Similar to CARLA, LGSVL simulator has a server-client architecture. The server side is where the simulation is run and the environment is loaded. The python API is used to control the simulation that runs on the server side. LGSVL simulator has a webUI in which the type of simulation can be chosen. There are different ways to run a simulation in the LGSVL. It can be run in *Headless mode* or using the python API. The method of python API was chosen to run the simulation since this provides direct access to the sensors and libraries of the simulator. This API also gives direct access to control the simulation. Consequently, a connection must be established between the client API and the server running the simulation. This is enabled by creating an instance of the **Simulator** class by specifying the IP address and the port. By default the IP address is the **"localhost"** and the port number is **"8181"**. Once this instance is created, then the maps and vehicles can be loaded.

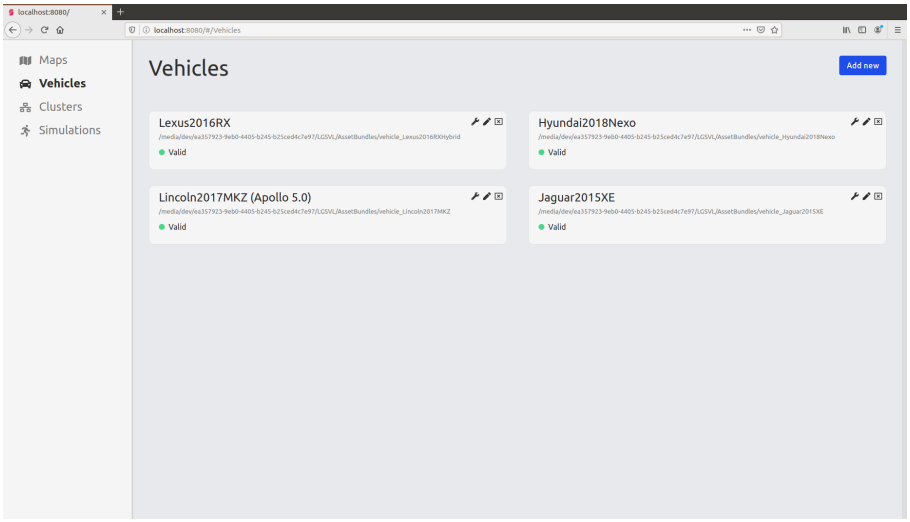


Figure 4.9: LGSVL WebUI for vehicles

The maps and vehicles are also loaded into the simulator by means of the webUI. A number of maps are available by default in the LGSVL simulator. From the default maps, **"Borregas Avenue"** is chosen as the map which is a Digital twin environment of a real world street. Likewise, several vehicles are available by default in the simulator. The vehicle model "Jaguar 2015 XE" was chosen as the choice of the vehicle. The choice of the vehicle is of not high importance in this thesis. Also, it is to be noted that currently there are no trucks available in the LGSVL simulator. Once the environment and the vehicles are loaded the necessary sensors need to be attached to the vehicles. As required, three vehicles are loaded in the simulator of which one will perform the cutin. LGSVL provides a wide array of sensors through its libraries. Some of the

sensors include Main camera, Depth camera, Radar, Lidar, CAN Bus, GPS sensor, Odometry sensor etc. However, for the current implementation Main Camera, Lidar, Radar, CAN Bus and GPS sensor are attached to the vehicle. These sensors are attached to the vehicle by means of the webUI. However, these sensors must be enabled through the client API to put the sensor in use and retrieve data from the sensor. As per the flow diagram, the vehicles in the platoon are initialised with the initial ACC velocity of 10 m/s. The *Target Vehicle* is initialised with a velocity of 15 m/s higher than the vehicles in the platoon.

Figure 4.10: LGSVL environment with vehicles

As opposed to the collision detection in CARLA, the collision detection in LGSVL is done by means of a default function in the API called **on_collision** and the vehicles are passed as parameters to detect the collision between them. Subsequently, the Cutin and the distance checking is performed in parallel. For controlling the basic actions of a vehicle, the LGSVL simulator has few functions in the API which helps with the control. LGSVL has a **VehicleControl** class which has a **apply_Control** function which helps in providing basic controls such as throttle, steering, reverse, braking and handbrake [43]. The implementation of the scenario makes use of this controller to perform the braking and accelerating actions of the vehicles. As opposed to a function returning the location of the vehicle during a tick in CARLA, LGSVL makes use of GPS sensor to get the location of the vehicle. The data from the GPS is both available as a location vector in the map and also as a GPS co-ordinates. This location is constantly monitored after the Cutin and if there the vehicles are within the ACC distance, appropriate braking action is taken to avoid collisions. The timeout can also be set in the simulator after which the simulation will timeout and exit.

Figure 4.11: Cutin Instant LGSVL

The effect of cutin in the LGSVL simulator is made evident by means of the velocity profile of the vehicles. The behavior of the vehicles is expected to behave as the CARLA simulator and from the velocity profile plot it can be seen that the cutin behavior is the same. After cutin, the *Target Vehicle*, which is represented as CUTIN in the plot, speed is reduced and comes in close proximity with the platoon. This causes the *Lead vehicle* in the platoon to perform braking action and come within ACC distance which triggers the *Trailing Vehicle*, which is represented as EGO, in the plot to brake. Once the vehicles are in safe distance from each other the vehicles return to their initial velocity. The behavior is captured in a velocity profile plot in the Figure 4.12:

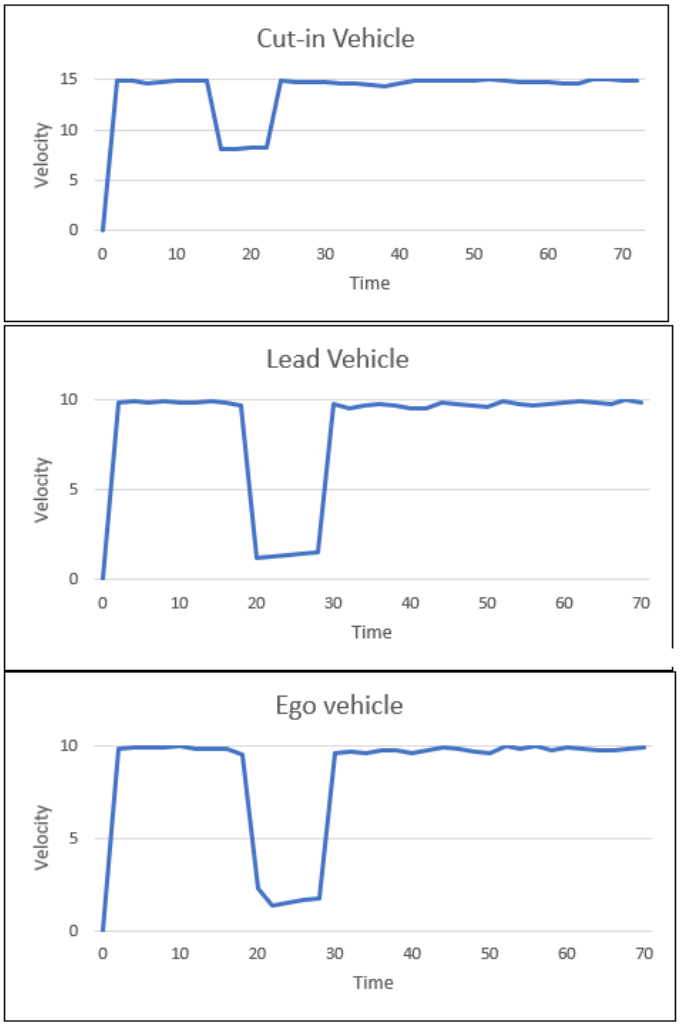


Figure 4.12: Velocity Profile plot - LGSVL

## 4.5 Cutin Scenario implementation in Gazebo

The Cutin Scenario implementation in Gazebo has a different implementation technique although the general flow of the code follows the flowchart in the Figure 4.3. Gazebo is a generic multi-robot simulator and not an autonomous vehicle simulator such as Carla and LGSVL. Therefore, Gazebo by default does not have all the libraries and the API that are required for the implementation of the cutin scenario. However, TNO currently uses Gazebo for their HIL simulation verification. Consequently, a custom made API is developed by TNO in Gazebo simulator. As mentioned before

the TNO HIL environment makes use of ROS as a middleware for communication. The API makes use of the ROS messages to conver them into commands which is then processed by the API and the corresponding action is taken. The ros messages are published under the topic **/gazebo/scenario/command** and this is used for the API interaction. A plugin is written in Gazebo which will process these messages and convert them into commands. These commands are then read and passed on to the appropriate controllers and processed which results in the user desired action.

The simulator can be launched with simple **"gazebo"** command from the terminal. However, in the HIL setup it is integrated with ROS. Thus, the gazebo simulator is launched along with the environment in which the vehicles will run. The map is not available by default as opposed to the other simulators. Instead, it is defined in OpenDRIVE and it is launched along with the simulator. There is no client-server topology in Gazebo. Similarly, the vehicles need to be launched in the simulator. The vehicle models are defined in a **urdf** file which is used to describe the robots. This consists of the vehicle dynamics and dimensions. The vehicle model of **"Toyota Prius"** is used as the vehicles for the scenario. The vehicles are launched by means of a launch file which loads the vehicles in the simulator. Three vehicles are launched for the current scenario simulation and they are launched in the map which consists of a long straight highway.

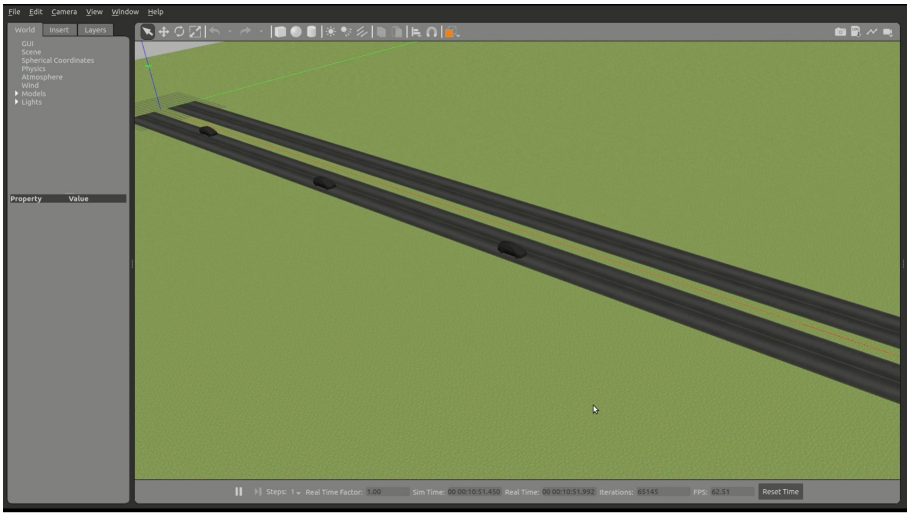


Figure 4.13: Gazebo simulation environment with vehicles

There are several different vehicle controller plugins written in Gazebo which are made use of to control the vehicle basic movements such as acceleration, deceleration and braking. Even for a lane change there is LaneChangeController plugin. The ros messages which are parsed as commands will call the respective controller with the arguments specified in the ros message. For instance, the message **"control modify controller lead SpeedController vx 10.0 ax 2.0"** will modify the speed of the lead vehicle in the platoon to 10 m/s and this makes use of the Speed Controller. First, the vehicles in the platoon are spawned in the right lane of the road and the Cutin vehicle is spawned in the left lane. The initial velocities of the platoon vehicles are set at ACC velocity. Unlike Carla and LGSVL there is no default function to return the location of the vehicle during a particular simulation tick. Therefore, a ros node was written to publish the pose message in a separate ros topic and a subscriber to that topic was written in the implementation. The pose messages were published under the topic **vehicle_name/gazebo/pose** where the vehicle_name denotes whether the vehicle is lead or trailing or the Cutin vehicle.

The lane change from the Cutin vehicle is also performed by means of the LaneChangeController plugin written in C++. In the other simulators the default vehicle controller can be used to control the vehicle steer and achieve the desired lane change manoeuvre. But in case of gazebo

there is no default controller in the simulator and hence a user-defined lane change controller is written which enables the lane change by specifying the timeframe and the distance. This is both advantageous and disadvantageous. Gazebo offers little functions by default but provides incredible flexibility to create many user plugins. However, there are a lot of plugins that need to be written for a huge application such as Truck platooning. The lane change is performed and the cutin vehicle comes in close proximity with the lead vehicle in the platoon. This triggers the braking action of the lead vehicle and subsequent braking action from the trailing vehicle.

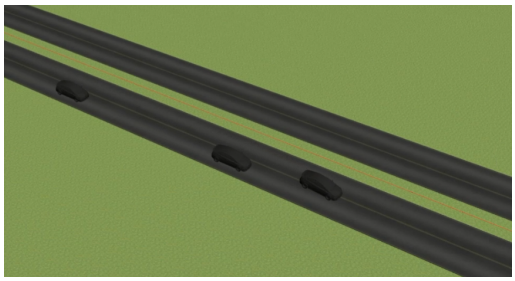

Figure 4.14: Cutin instant Gazebo

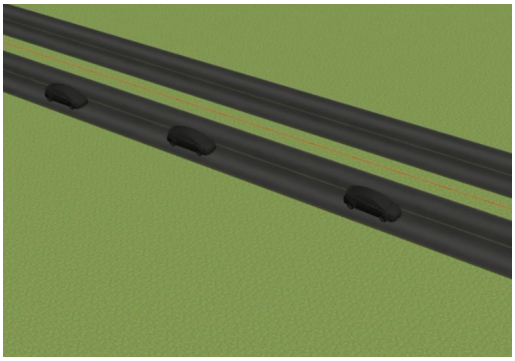

Figure 4.15: After cutin Gazebo

The effect of the cutin is also graphically visualised by means of the velocity profile plot which shows the braking actions of the vehicles in the platoon because of the cutin in front of the lead vehicle. The velocity of the vehicles are also obtained by following a similar approach like the location. The velocities are constantly published for every simulation tick via rostopic and the rostopic is subscribed through a subscriber to read the velocity.

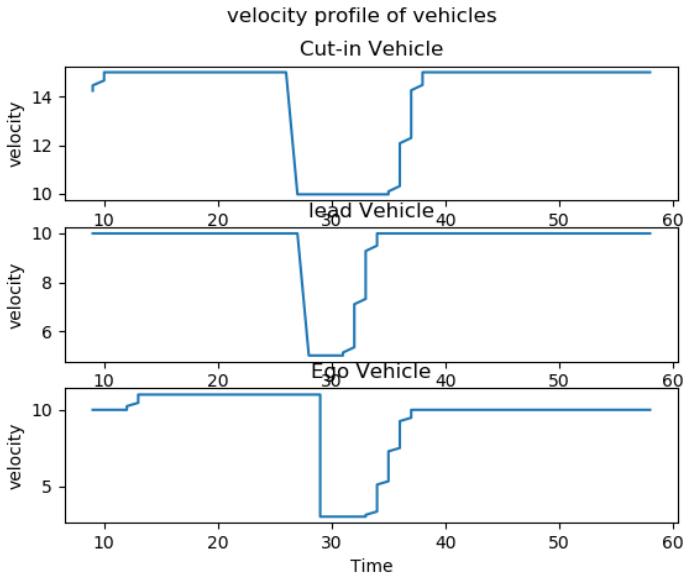


Figure 4.16: Velocity Profile plot - Gazebo

As expected, the velocity profile plot in the Figure 4.16 clearly shows the effect of cutin on the vehicles in the platoon. After the cutin, there can be a drop in velocity observed in the vehicles of the platoon since they are in ACC and once the cutin vehicle is within safe distance from the platoon, the vehicles again restore to their usual behaviour.

Based on these implementation details and the possibilities of introducing V2V communication and large scale traffic, the three simulators will be analysed in detail and will be evaluated based on the Application specific KPIs. This results in an effective simulator which is explained in detail in the next chapter.

# Chapter 5

# Analysis and Evaluation of Simulators

The implementation of the Cutin scenario in front of the lead vehicle in the three simulator was detailed in the Chapter 4. To find the effective simulator for the truck platooning application and for the Application specific key performance Indicators, the simulators have to be evaluated based on the implementations. This chapter explains in detail the mapping of the scenario to steps, the analysis of the simulators for the specific KPIs and the evaluation method of the simulators. Finally, an effective simulator for the truck platooning application is found. There is no "one for all" simulator or one ideal simulator. Therefore, the thesis aims to provide an effective simulator for the application which can be used based on the requirements of the project.

## 5.1 Evaluation Approach

The Cutin scenario was separated into steps to create the abstraction between the common scenario events and the implementation details in the simulators, which would help analyse the simulators based on the features of the simulator. There should be an evaluation approach to benchmark the simulators with the Use Case. The approach should pave the way to quantify and validate the non-quantifiable scenario implementation details and simulator capabilities of different simulators. Therefore, the decided approach consists of mapping the Key performance Indicators to the scenario steps and performing a detailed analysis of them in each of the simulators. Subsequently, based on the analysis, the simulators will be scored on a scoring system and the effective simulator will be found for the respective KPIs.

The scenario steps, as illustrated in the Figure 4.2, are the common steps for all the simulators. These are akin to the abstract layer where the underlying implementation details in the simulation will differ. The underlying implementation details of the simulators will be evaluated based on the default features and availability, possible extension features i.e. Plugin Creation and Flexibility, Integration capabilities of the simulators with other simulators - because some application or projects demand the use of multiple simulators and Lines of code of the implementation in the simulators. These parameters form the base for the scoring system. The scoring system revolves around the analysis of the simulators based on the above mentioned aspects. As a result, an appropriate score is assigned for the simulator for a particular KPI. Moreover, in addition to the application specific KPIs, there are few other KPIs that are relevant to the scenario steps. A brief analysis of those relevant KPIs are also considered for the evaluation of the simulators. The scoring system is a numerical rating system based on the analysis of the simulator with respect to the above mentioned scoring parameters. The scoring system will be explained in detail in Section 5.3.

## 5.2 Mapping KPIs to Scenario steps

The first step in the evaluation is to map the KPIs that are selected and applicable to the steps of the scenario. These steps describe the common aspects for all the simulator. Hence, the mapping of the scenario steps to the KPIs depicts the relevance of the selected Use Case for the evaluation of the simulators. If the KPIs were not be able to map to the scenario steps then the chosen scenario is irrelevant and thus invalid for the analysis of the simulators. So mapping the KPIs to the scenario steps is an important step in answering the second research question RQ2.

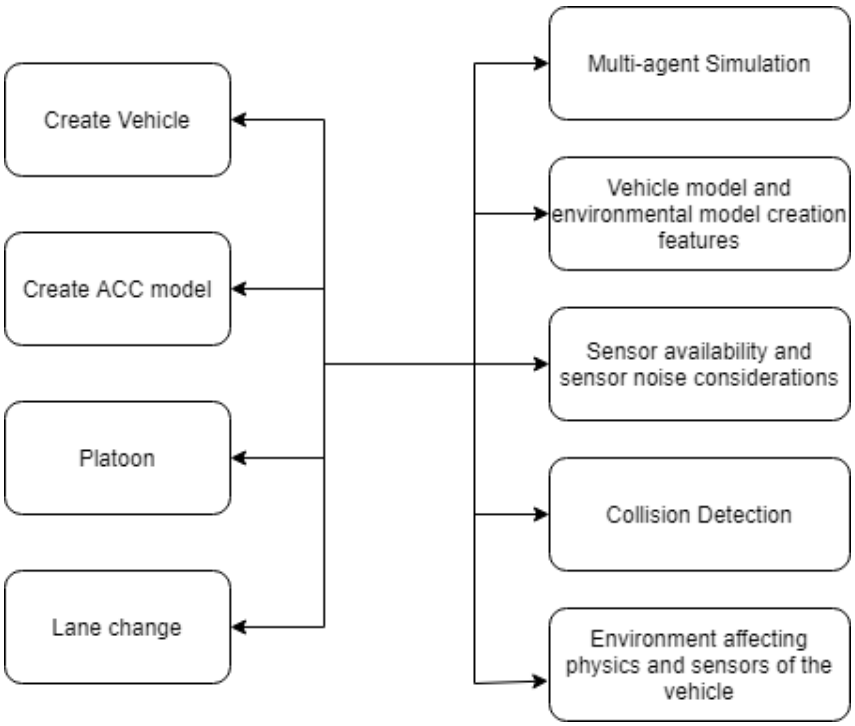


Figure 5.1: Mapping of KPIs to Scenario steps - 1

It can be observed from the Figure 5.1 that the steps *Create Vehicle*, *Create ACC model*, *Platoon* and *Lane change* can be mapped onto five Key performance Indicators. Although, the emphasis is placed on the Application specific KPI - Multi-agent Simulation, all the five KPIs can be mapped to these scenario steps. Truck platooning application is a multi-agent system in itself. For the chosen scenario three vehicles have to be created and the vehicles need to interact with the other vehicles and with the environment, which falls under the definition of multi-agent system as explained in [44]. The vehicle model and environment models that are used in the implemented scenario are the default models that are used in the simulator. However, the prospects and the capabilities of the simulator in defining new vehicle models need to be analysed. Therefore, this KPI is mapped to the scenario component. The sensors are required to read the data from the environment and the vehicle and are also used to make actuator perform certain actions based on the data. Therefore, the sensor availability and noise considerations has a relevance for this scenario component. Similarly, for the ACC model collision detection is a must have KPI because the main goal of the ACC system is to avoid collision and the simulator should detect the collisions. The final KPI, which is the environment affecting the physics and sensors of the vehicle, is also important for the scenario component. Although, it is not a must have KPI for the basic scenario and rather a nice to have KPI, but for for more realistic simulations it is a must have KPI.

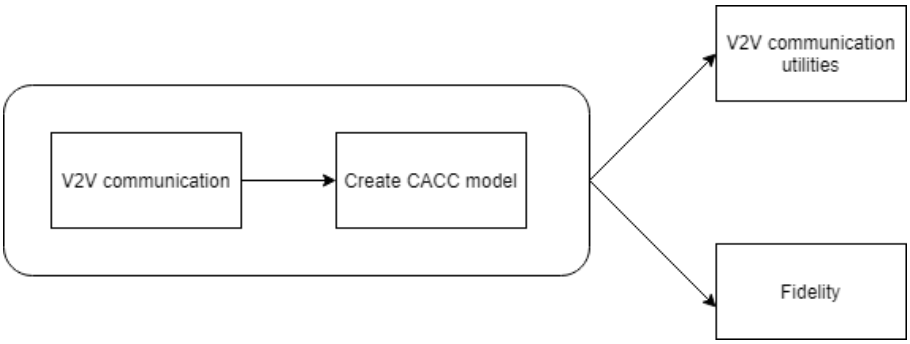


Figure 5.2: Mapping of KPIs to Scenario steps - 2

The second category of mapping KPIs to scenario steps involve the steps *Create CACC model* and *V2V communication* as depicted in the Figure 5.2. The mapping of the KPI for this category is rather straightforward. The *Create CACC model*, which involves the V2V communication, has a direct mapping to the KPI V2V communication utilities. This mapping paves way for the analysis of the simulator with respect to the capabilities of the simulator to simulate the Vehicle to vehicle communication. Also, Fidelity can be a good additional mapping to this category. Fidelity in this context means the consideration of noise and signal loss while transmission of data during the V2V communication. This fidelity represents the realism in the communication. Fidelity in this case considers the possibilities of introducing noise and path loss models in the communication channel. Therefore, these two KPIs are relevant for this category of scenario steps.

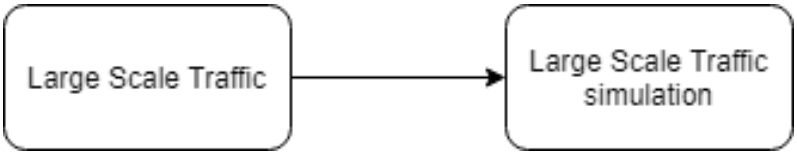


Figure 5.3: Mapping of KPIs to Scenario steps - 3

The third mapping category is concerned with the component *Large Scale Traffic* as shown in the Figure 5.3. This mapping is concerned with the ability of the simulator to generate large scale traffic and handle large scale traffic. This mapping is also rather straightforward mapping from the KPI to the scenario component. Although the second and third categories of the scenario steps and their respective mapped KPIs are not implemented, the implementation and simulator capabilities are analysed and evaluated.

## 5.3 Scoring System

The first step for the evaluation for the analysis and evaluation of the simulators is the mapping of KPIs to the steps. Then the analysis of the simulators have to be performed based on the KPIs and scenario implementation in the simulators. Finally, a scoring system is decided and each simulator is scored for a particular KPI based on some scoring parameters. This gives a score for each simulator for each Application specific aspect KPI and final results are achieved. The analysis and scoring are based on certain scoring parameters. As mentioned before, the scenario steps are the abstraction layer. These specifications are the common layer which all the simulators must implement. The scenario steps could be specified using a common Domain Specific Language, for instance OpenScenario, and only the underlying implementation details of the simulators such as Gazebo, CARLA and LGSVL vary. The scenario steps can be specified as events in the storyboard of the OpenScenario but the implementation of ACC, V2V communication could be different in the simulators. Hence, the scoring parameters are chosen which describe the capabilities and

possibilities of the simulator in implementing a particular KPI and also possible integration and extension capabilities. Therefore, three scoring parameters are designated based on which the simulators will be rated for each of the KPI if it is applicable. The three scoring parameters are **Default Features and Availability**, **Extension Capabilities** and **Integration with other simulators**. Each of this parameter has a maximum score of **1.0** and every simulator will be rated based on the scoring parameter between 0.0 to 1.0. The simulators can have a maximum score of **3.0** for a KPI and a minimum score of **0.0** for a KPI. The simulator with the highest score for each KPI is the effective simulator for the respective KPI. The overall highest score of the simulator indicates that the simulator is the effective among the three simulators for the Truck platooning application.

## 5.4 V2X Communication Utilities

### 5.4.1 Gazebo

Gazebo is a 3D robotics simulator that can simulate robots and can be also used for autonomous agent simulations. The other simulators under consideration, CARLA and LGSVL, are specific for autonomous driving simulation whereas the Gazebo simulator is a general robotic simulator. Therefore, it offers much flexibility but also demands a lot of tooling to be developed to achieve desired goals. Moreover, it has extensive support for ROS which makes the integration with ROS easier. Thus, Gazebo in addition to having default sensors which enable the transmission and reception, it can also be integrated with ROS environment to enable communication.

**Default Features and Availability:** Gazebo simulator has a Sensor class which provides sensors that supports transmission and reception of wireless signals [45]. This can be made use of in establishing the communication between the vehicles. Gazebo offers three sensors that helps in this case. They are Wireless Transmitter sensor, Wireless Receiver sensor and Wireless transceiver sensor. The sensor plugins which are available provide skeleton of code for these sensors which can then be exploited to the user needs. For instance, the transmitter sensor provides interface to define the obstacles between the transmission and receiver, it defines a free space propagation model for the signals etc [46]. Similarly, on the receiver side the receiver sensitivity and the receiver gain can be fine tuned. There is also possibility to introduce noise to the sensors. These features make Gazebo a great tool to simulate the the Vehicle to vehicle communication. Based on the above aspects, the score for Gazebo in this category is **1.0**.

**Extension Capabilities:** There is also another method by which the V2V communication can be established in Gazebo. As mentioned before, Gazebo simulator has excellent integration capabilities with ROS. The ROS node **relay** can be used to simulate the vehicle to vehicle communication in the Gazebo simulator [47]. The ROS relay is a node that can be used to subscribe to a ros topic and republish all the data from the subscribed topic to another topic. In this manner the lead vehicle can publish to and subscribe from the data of the trailing vehicle and vice versa. The only disadvantage of this method is that it assumes that the communication is always ideal. There will not be any data loss but in reality there is noise and path loss in transmission. However, for the simulation purpose this method proves to be a useful manner for verification of V2V model and in the simulation of CACC model. The score for the Gazebo simulator in this category is **1.0.**

**Integration with other simulators:** Vehicle to Vehicle communication is a huge topic by itself which demands a lot of emphasis. There are several network simulators that simulate the effect of Vehicle to vehicle communication which includes the path loss models, latency, packet delivery ratio etc. Some of the simulators that can simulate these aspects are OMNet++, ns-3 etc. However, these simulators are majorly focussed on the technical aspects and efficiency of the V2V protocol that is being used and reliability aspects.

The integration of Gazebo simulator with these simulators will not be so effective since the

focus of the truck platooning application is not entirely based on V2V communication. The integration is also a difficult task. Moreover, the thesis is not focused on the efficiency of V2V communication but rather the truck platooning application in general. Therefore, the integration with network simulators is **Not Applicable** for this category.

### 5.4.2 CARLA

CARLA is an autonomous driving simulator which has several default features of autonomous driving and Advanced driver assistance system capabilities embedded in its simulation environment. However, the V2V communication and V2X communication capabilities are not readily available. Therefore, if one needs to implement a V2V infrastructure in the simulator the open source features of the capabilities have to be explore i.e. the possibilities of extending the simulator to add additional tooling to enable V2V infrastructure need to be explored.

**Default Features and Availability:** As mentioned before the CARLA simulator by default does not provide any infrastructure for V2V and V2X communication. For the implementation of the ACC model the location of the vehicle was constantly read and decisions are taken. However, if the Cutin scenario were to be implemented completely, then a V2V is need to create the CACC model which the CARLA simulator lacks. However, the simulator is being constantly upgraded with every version. Since there is no default feature of the simulator to simulate the V2V communication, the score for this simulator in this category is **0.0**.

**Extension Capabilities:** Since the simulator does not provide any default support for the V2V communication, there is a need of flexibility in the simulator so that it can be extended to support the V2V and V2X communication capabilities. CARLA simulator has also good support for integration with ROS environment [48]. This makes the CARLA simulator extensible for implementing the V2V communication. A ROS bridge has to be installed in the CARLA simulation environment which enables integration with ROS. This facilitates subscribing and publishing to ros topics. Therefore, the concept of ROS relay can be used to implement the V2V communication. Since the CARLA has extension features that can be used for the V2V communication, the score of this simulator in this category is **1.0.**

**Integration with Other simulators:** Although CARLA simulators several integration possibilities with the other simulators. But as indicated in the analysis of Gazebo simulator the integration is **Not Applicable** for this thesis.

### 5.4.3 LGSVL

Similar to CARLA simulator, LGSVL simulator is also an autonomous driving simulator. It is mainly used for simulating autonomous vehicle and providing simulation capabilities of Advanced Driver Assistance Systems. It also has simulation capabilities of implementing Artificial Intelligence and Machine learning in the simulation environment. Moreover, it is also capable of generating HD maps. However, the V2V communication aspects of the simulator is a bit limited. The simulator offers limited support to the V2V communication and truck platooning application. This is discussed in detail below:

**Default Features and Availability:** The LGSVL simulator by default does not support the simulation of the vehicle to vehicle communication or the vehicle to infrastructure communication. Although it offers several default sensors, it does not provide any sensor infrastructure like Gazebo to facilitate the simulation of V2V communication. However, it is also to be noted that the simulator is being upgraded often thus in the future there might be an infrastructure or sensor

class to support the V2V communication.

**Extension Capabilities:** The LGSVL simulator by default does not offer any V2V support. Therefore, the simulator's extension and tooling capabilities have to be analysed to provide support for the V2V communication. On analysis, it is found that the simulator offers good integration capabilities with other software for self-driving vehicles such as Autoware AI, Apollo 3.0 and Apollo 5.0. Moreover, it offers good integration support with both ROS and ROS2.

Similar to CARLA, the LGSVL simulator has a bridge through which ROS can be integrated with the LGSVL environment. Then the ROS relay node can be used to establish the communication between the vehicles in the platoon. LGSVL has a GPS sensor which provides the location of the vehicle and a CAN bus sensor which provides details of the vehicle such as speed, braking, throttle values. The data from these sensors can be published to their respective ros topics. These topics can be published and the subscribed to and republished, to establish communication using ROS relay. Based on this analysis, the score of the simulator in this category is **1.0.**

**Integration with Other simulators:** LGSVL simulators several integration possibilities with the other simulators. But as indicated in the analysis of Gazebo simulator the integration is **Not Applicable** for this thesis.

## 5.5 Large Scale Traffic Simulation

Large-scale traffic simulation is the ability of the simulator to simulate realistic urban traffic scenarios for testing various automotive scenarios. This large scale traffic generation is useful for urban planning, road network design in addition to the realism in autonomous vehicle testing. Three types of simulation techniques are used in modelling traffic flow simulation [49] : First is the Agent-based simulation or microscopic methods where the motion of each vehicle individually is determined through a set of rules. The rules can be easily varied on a car-car basis. On the other hand is the Continuum or macroscopic approach which describe the motion of many vehicles with aggregated behavior. Third is the mesoscopic traffic model which focuses on vehicle groups. Agent-based simulation captures individual vehicle behavior while the macroscopic method is focused on efficiency. In literature there are few researches which even proposes hybrid traffic simulation [49]. One well known traffic simulator namely PTV Vissim [50] can majorly support microscopic traffic flow simulation with additional support for all methods of traffic simulation.

In this dissertation, the focus is placed on the simulator's capabilities to simulate and add large scale traffic in the simulation environment. There should be features to control the vehicles in the simulation that are in the traffic. The text following this will be explaining the traffic simulation capabilities and support from which a suitable simulator can be decided.

### 5.5.1 CARLA

**Default Features and Availability:** CARLA simulator has a sophisticated entity called Traffic Manager (TM) which is responsible for controlling the traffic and vehicles inside the simulation. The traffic manager populates the simulation with realistic urban traffic conditions. This is made possible by controlling the registered vehicles by setting their autopilot to True. The TM is on the client side of Carla and has different stages which enable the traffic control. These stages usually run on a different thread[51].

Moreover, the traffic flow can be customized by setting the parameters through the software or the client API. Once can force a lane change or ignore speed limits thus making the traffic simulation close to reality. The various stages involved in the Traffic Manager is as follows:
**Localization stage:** Responsible for the storage of waypoints for the vehicles to follow and updated accordingly to the vehicle behavior.

**Collision Stage:** This checks for the possible collisions for all the vehicle. This stage determines the priority of vehicle and communicates it to the Motion planner stage.
**Traffic light stage:** All distance between vehicles are determined by a geodesic boundary and collision or traffic violation is triggered when one vehicle enters the other's boundary. This stage manages the traffic regulations.
**Motion planner stage:** This stages combines information from the above stages and makes the decision on the movement of the vehicles. It uses PID controller to adjust the vehicles behavior.
**Apply Control Stage:** This applies the actuation signals from the motion planner stage such as throttle, brake, steer etc to the vehicles.

Also, from the large scale traffic model described above it can be observed that the CARLA has a hybrid traffic flow simulation strategy with more inclination towards the mesoscopic simulation method. In general a **Carla.Client** can be used to create a TM instance connected to a port. For safety purposes a minimum distance between the stopped vehicles can be set and also an intended speed can be set. Additionally there is also possibility to set/reset traffic lights. User can enable/disable collision between a vehicle and an actor and make a vehicle ignore other entities. Also forcible lane changes can be enabled or disabled. Additionally the hybrid physics mode can be also set. Hybrid physics mode is where the physics of the vehicle can be enabled or disabled when approaching the ego vehicle at a particular radius. Based on the default supports that CARLA provides for the large scale simulation, the simulator is assigned a score of **1.0** for this category.

**Extension Capabilities:** CARLA simulator by default provides urban traffic generation in its simulation environment. Therefore, there is no need of extension tools in the simulator to support the large scale traffic. Also, it supports additional integration with other traffic simulators. The simulator itself cannot be extended, however the existing traffic manager can be exploited to the needs of the user. Therefore, this category gets a score of **1.0**.

**Integration with other simulators:** CARLA simulator supports good integration with other traffic simulators, in addition to its own traffic manager. Carla also supports co-simulation with other traffic simulators such as PTV Vissim and Sumo, albeit at a basic level. This makes Carla a great simulation tool for generating urban traffic. Both PTV Vissim and SUMO are well known micro traffic flow simulators. Although the support for integration is at a basic level, with every update the support is being increased for the co-simulation. The integration support can be extended with software experts since the CARLA simulation software is open source and we can extend it to our advantage. Since it offers good support for integration with other traffic simulators, the score for CARLA in this category is **1.0.**

### 5.5.2 LGSVL

**Default Features and Availability:** LGSVL simulator does not have a Traffic manager by default like CARLA to simulate realistic urban traffic scenarios. However, it provides the feasibility of the simulating large scale traffic by means of agents called as Non-player Characters(NPC) vehicles and pedestrians. There are several different NPC vehicles such as Sedan, SUV, Boxtrucks etc and also different pedestrian types. These NPC vehicles can be made to drive along the lanes or a specific waypoints on the map. Moreover, there are several functionalities that NPC vehicles can be made to do which facilitates the large scale traffic simulation. It has the functionalities such as Change lanes, Follow specific waypoints, follow lanes and methods to set call back function when vehicle reaches a stop line at the intersection or changes lane. Moreover, it is to be noted that the NPC vehicles ignore the traffic rules and will not avoid collisions to get to the next waypoint. These addition of NPC vehicles to the traffic can be made by means of the client API [43].

Another way is to add a pool of NPC vehicles to the simulation which will behave in the same way as NPCs added to a non-API simulation. In this case, the vehicles will follow the map

annotations, obey speed limits, traffic signals etc. However, the user cannot control these vehicles. Similarly, the pedestrian behavior can also be simulated. They can be made to walk randomly or follow specific waypoints. This addition is similar to the NPC vehicles. These features make the simulator good for the large scale traffic simulation. As can be observed, LGSVL facilitates only Agent based or microscopic simulation where individual agents can be controlled. If no control is needed then one can make use of the option where the NPC vehicles cannot be controlled. However, CARLA has a special entity to handle the large scale traffic but LGSVL has no manager to control the traffic. As the number of vehicle increases, it is difficult to handle and control the traffic load in LGSVL simulator. Based on the analysis, the simulator can be assigned a score of **1.0.**

**Extension Capabilities:** The simulator has no entity that can be extended to add additional support to large scale traffic simulation. Although an entity similar to CARLA can be created, but it involves a long time and resource investment which is not desirable. Therefore, the simulator gets a score of **0.0** for this category.

**Integration with other simulators:** LGSVL, although it supports good integration with other softwares such as ROS, Apollo, it does not have any support for large scale traffic simulators unlike CARLA. An approach similar to the one adopted in CARLA can be used to make the LGSVL support the integration with other simulators but it consumes lot of time and resource. Hence, the score for this simulator in this category is **0.0**

### 5.5.3 Gazebo

**Default Features and Availability:** Gazebo by itself does not support any traffic simulation since it is a robotic simulator and not an autonomous driver simulator. The ways in which traffic can be created is by adding multiple vehicles and defining a route to follow which is really complex and difficult to manager. The other way is to integrate with the other micro traffic flow simulators such as Sumo and PTV Vissim. Therefore, for large scale simulator Gazebo simulator individually cannot be used. The score for the simulator in this category is **0.0**.

**Extension Capabilities:** As mentioned before, the Gazebo simulator does not offer any tooling or an entity like traffic manager to support large scale traffic. Multiple agents can be added but the control of all the agents will be really difficult if the traffic is large scale. Thus, the simulator cannot be extended to support large scale traffic. The score of the simulator for this category is **0.0**.

**Integration with other simulators:** Although Gazebo does not offer any support for Large scale traffic simulator, it can be integrated with the other traffic simulators to support large scale traffic. This is also being done in the literature in [52]. In [52], a hybrid simulation tool is proposed by integrating SUMO simulator for large scale traffic and Gazebo. This is enabled by the use of ROS. Similarly, the tooling can be written to integrate the Gazebo simulator with the PTV Vissim. The approach similar to the one employed in CARLA can be used to develop a tooling to integrate it with PTV Vissim. Since the integration with traffic simulators is a feasibility, the simulator can be assigned a score of **1.0.**

## 5.6 Multi-agent Simulation

Multi-agent simulation is the ability of the simulator to handle and control multiple agents in the simulation environment. The simulator possess the ability to create multiple agents, add features, control and handle them to perform desired actions. It is a simulation where multiple autonomous agents interacting in the same simulation. The reason for the multi-agent simulation

could be to test how agents react to one another, but also enable testing their cooperative and competitive behaviour. In the context of truck platooning, we need to test their cooperative behaviour. Therefore, the analysis of multi-agent simulation capabilities is important.

### 5.6.1 CARLA

**Default Features and availability:** As observed from the Figure 5.1, the multi-agent simulation can be mapped to the four scenario steps. For the Create Vehicle step, CARLA by default provides several vehicles that can be used to create simulations. For instance, CARLA has several vehicle blueprints that have the size and scale of the actual cars. Few of the vehicles provided by CARLA include Audi A2, Tesla model 3, Toyota Prius etc. [53]. CARLA uses default vehicle model which is used by Unreal Engine. The Unreal Engine uses NVIDIA PhysX vehicle model for the modelling of vehicle and dynamics. Although, there are several cars, two-wheelers and pedestrians which are available in CARLA, there are no trucks provided by default. Since trucks are required for the platooning application, the simulator is expected to have the trucks by default.

For the remaining scenario steps in the Figure 5.1, the sensors and Vehicle control play a major role. Since the interaction of the autonomous agents and the environment is an important feature of the multi-agent simulation, the availability of sensors and the vehicle controllers are analysed. CARLA offers a wide array of sensors such as Radar, LIDAR, Odometry, GNSS sensors etc. For the implementation of Truck platooning vehicle controllers play an important role. The controllers are linked to the dynamics of the vehicle. CARLA has a class Vehicle.ApplyControl which is the native vehicle controller for CARLA. The values of throttle, steer, braking, reverse, handbraking and reverse can be provided by passing the appropriate values of these parameters through an instance of the class. The values are then passed to the PID controller in the CARLA which enables and controls the movement of the vehicles. This makes the API of the CARLA easy to use and provides good abstraction from the underlying details. This controller was used to perform the lane change and control the movement of the vehicles in the current implementation. Additionally, through the API the physics of the vehicle can also be modified which is also a good abstraction instead of modifying the physics through the vehicle model. Overall, the default support for the multi-agent simulation in CARLA is really convenient for the truck platooning application and thus it is assigned a score of **1.0**.

**Extension Capabilities:** The extension capabilities of the multi-agent simulation concerns with the adding more vehicle models, modelling a vehicle, adding sensor plugins and user defined controllers. CARLA is both open-source and offers flexibility with extensions. It provides interfaces where new sensors can be written and integrated. Moreover, new environments can also be created with OpenDRIVE and can be integrated into the simulation environment. As for modelling the vehicle, two ways are possible. A default skeleton is provided in the CARLA environment for four wheeled vehicles in its repository from which vehicles can be modelled and extended [54]. The created vehicle can then be imported in the CARLA blueprint library and can be used. Another method is to create a urdf vehicle model, for instance trucks, and then import it using ROS. Since the simulator is open-source we can also define our own sensors and controllers. However, these additions can be done only necessary since it offers good default features. Therefore, CARLA offers good extension capabilities in addition to its default support. Based on this analysis, a score of **1.0** is assigned to CARLA for this category.

**Integration with other simulators:** CARLA offers several integration interfaces with other simulators, such as SUMO and PTV VISSIM. The integration of CARLA with simulators such as SUMO and VISSIM adds capability to CARLA to have multiple agents in the simulation environment and also can be used for Artificial Intelligence and learning. Additionally, it also offers good support for OpenSCENARIO and OpenDRIVE with which variety of maps can be injected and scenario based testing can be done. This support of OpenSCENARIO and OpenDRIVE makes it a suitable simulator for HIL testing. Considering this analysis, the simulator can be assigned a score of **1.0** for this category.

### 5.6.2 LGSVL

**Default Features and Availability:** LGSVL also offers a simulation environment with several default features for multi-agent simulation because it is a simulator which is focused on simulating autonomous vehicles and the vehicles has to interact with other vehicles and also the environment. Considering this for the first scenario step of *Create Vehicle*, LGSVL provides several vehicle models of actual cars. Currently, LGSVL supports five vehicles which are a simulation model of actual cars. All these vehicles are EGO vehicles which means that they can be manipulated and controlled through the simulation. LGSVL uses the default dynamics model which is a C based model which makes use of the Unity game engine's PhysX physics engine and the steps of it [55]. However, it also supports multiple dynamics models. The controller input is fed to the model and applied to the wheel colliders enabling the appropriate action of the vehicle. Additionally, it also provides two interfaces where the user can create their own dynamics model using their Simple model interface and Full model interface.

In addition to the EGO vehicles, there are also other NPC vehicles with simplified physics that can be added as vehicles for traffic. The NPC vehicles can perform few manoeuvres such as lane changing, following waypoints etc. Moreover, the available NPC vehicles are Sedan, SUV, Hatchback, Boxtruck, schoolbus and Jeep [56]. Thus, the agents that can be added in the simulation are plenty.

For the remaining steps of the scenario sensors and controllers play a major role. The sensor model provided by the LGSVL are plenty. It provides sensors such as CAN bus, GPS, Odometry, Radar, Lidar etc. [56]. Since LGSVL has integration with other AD stacks such as Apollo 3.0, Apollo 5.0, Autoware and also with ROS/ROS2, there is a separate bridge with which the sensors can be configured for each of the stacks and the sensor data can be published as a topic. Also, similar to CARLA, LGSVL allows users to create their own sensor plugins.

LGSVL has a **apply_control** method that can be used to control the vehicle actions. The values are passed as arguments to the functions which are passed to the Controller which enables the appropriate action of the vehicle. This controller is also a PID controller which controls the vehicle. The default features offered by LGSVL is suitable for a multi-agent simulation and thus a score of **1.0** can be assigned to this simulator.

**Extension Capabilities:** LGSVL is both open-source and also supports good extension capabilities of its software. For multi-agent simulation the extensions are focused on the creation of other vehicle models, sensor plugins and controllers. For adding a new vehicle, such as trucks, into the simulation environment, the Unity game engine can be used to create the EGO vehicle and its corresponding default vehicle dynamics [57]. The scripts that enables to create this new vehicle is provided by LGSVL in its repository. Additionally, it also offers a interface where they dynamics model of the vehicle can be modified which is important for truck platooning since the dynamics of the truck will be different from those of the cars. Similarly, for sensors a sensor plugin can be added and the bridges that it requires to connect with the other stacks can be added. From this analysis, it can be observed that LGSVL has good support for the extension of the simulator for multi-agent simulation. Hence, it can be assigned a score of **1.0** for this category.

**Integration with other simulators:** Although LGSVL supports integration with Autonomous Driving stacks such as Apollo, Autoware and also with ROS and ROS2, it does not support integration with other autonomous driving simulators to extend its support for multi-agent simulation. The multi-agent simulation can be performed with the existing simulator capabilities and by extending it. However, if a certain project demands integration of the simulator with other simulators, then the user has to define their own tooling to facilitate the integration. Therefore, the simulator is assigned a score of **0.0**.

### 5.6.3 Gazebo

**Default Features and Availability:** Gazebo simulator is a general purpose robotic simulator which is capable of simulating and controlling multiple robots. Therefore, the potential for supporting the multi-agent simulation in Gazebo is high. However, the support for multi-agent simulation in the context of truck platooning is rather limited because of its general purpose attribute. The main difference between Gazebo and the other two simulators in comparison is the provision of an API in CARLA and LGSVL. The API provides abstraction and access to the simulators' libraries and plugins. Although Gazebo provides few default robots for simulation and template classes for various sensors, the user need to configure the sensors according to their needs. This offers good flexibility and the libraries can be tailor made according to the user needs but can also be problematic and complex. For the *Create Vehicle* step, the vehicles are modelled in a Unified Robot Description Format(URDF) [58] which is a XML format in which the vehicle specifications can be specified. It contains all elements of the robot and used in ROS. The vehicle is created through this format and then launched into the simulation environment. The vehicle type that is used in the current implementation is the Toyota Prius model. In this manner, trucks can also be created proving the flexibility in Gazebo. However, there is a need of initial development of the required urdf file. In the current HIL setup of TNO, there is an API which was created to parse ros messages and modify them into commands which enables the required action in the simulation.

For the remaining scenario steps, which involves the sensors and the controllers, Gazebo requires again many user defined sensors and controllers. Since the steps involve lane change and performing Cutin different controllers are required to set velocity, perform lane change etc. Hence, there are several different controllers which are created in the API such as speed controller, lane change controller, acceleration controller. Moreover, ROS nodes are implemented in the current implementation to publish the **pose** of the vehicle to enable the ACC between the vehicles in the platoon. As it can be observed, the Gazebo offers certain default features required for the multi-agent simulation, but it also requires a lot of additional developments that needs to be implemented for the truck platooning application. Hence, the simulator is assigned a score of **0.5** for this category.

**Extension Capabilities:** The usability of the Gazebo simulator for the truck platooning application and multi-agent simulation in general is highly dependent on its extension capabilities. Gazebo also provides a platform where the simulator can be extended to satisfy the demands of the applications. For instance, it provides a Sensor class with a large collection of default sensor classes and methods [45]. The sensor class can then be modified and configured according to the user needs. Similarly, the integration with ROS enables Gazebo to support the Unified Robot Description Format(URDF) robots with which the vehicles can be integrated and simulated. Moreover, it also supports the integration with OpenSCENARIO and OpenDRIVE. However, the user has to write the complete tooling to extend its support for the file formats. Thus, extending the simulator for multi-agent simulation is possible in Gazebo but it consumes a lot of time for the development of tooling in comparison with other simulators. Thus, the simulator is assigned a score of **1.0** for this category.

**Integration with other simulators:** Although Gazebo supports the integration with ROS, the integration capabilities of the simulator with other simulators for the multi-agent simulation is minimal. For instance, if one has to integrate the Gazebo simulator with SUMO simulator for a co-simulation, the entire tooling has to be written by the user. A similar approach used by the CARLA simulator for co-simulation can be used for creating the integration tool. Thus, the possibility exists to integrate the simulator but it consumes time and resources to develop the tooling for the integration which is not highly desirable. Therefore, the simulator is assigned a score of **0.0** for this category.

## 5.7 Final Results

| Multi-agent Simulation | | | |
|---|---|---|---|
| **Scoring Parameters** | **CARLA** | **LGSVL** | **Gazebo** |
| Default Features and Availability | 1.0 | 1.0 | 0.5 |
| Extension Capabilities | 1.0 | 1.0 | 1.0 |
| Integration with other simulators | 1.0 | 0.0 | 0.0 |

Table 5.1: Multi-agent Simulation Evaluation

As observed from the Table 5.1, for the multi-agent simulation CARLA simulator is more suitable for the truck platooning application. From the analysis of the simulator for the multi-agent simulation capabilities, it achieves the highest score among the simulators. Therefore, if the project demands focus on multi-agent simulation, the CARLA simulator would be the effective simulator.

| Large Scale Traffic Simulation | | | |
|---|---|---|---|
| **Scoring Parameters** | **CARLA** | **LGSVL** | **Gazebo** |
| Default Features and Availability | 1.0 | 1.0 | 0.0 |
| Extension Capabilities | 1.0 | 0.0 | 0.0 |
| Integration with other simulators | 1.0 | 0.0 | 1.0 |

Table 5.2: Large Scale Traffic Simulation Evaluation

As observed from the Table 5.2, for the large scale traffic simulation CARLA simulator is more suitable for the truck platooning application. From the analysis of the simulator for the large scale traffic simulation capabilities, it achieves the highest score among the simulators. It offers default support for the large scale traffic simulation and also integration capabilities with other simulators. Therefore, if the project demands focus on large scale traffic simulation, the CARLA simulator would be the effective simulator.

| V2X Communication Utilities | | | |
|---|---|---|---|
| **Scoring Parameters** | **CARLA** | **LGSVL** | **Gazebo** |
| Default Features and Availability | 0.0 | 0.0 | 1.0 |
| Extension Capabilities | 1.0 | 1.0 | 1.0 |
| Integration with other simulators | NA | NA | NA |

Table 5.3: V2X Communication Utilities Evaluation

As observed from the Table 5.3, for the V2X communication utilities simulation Gazebo simulator is more suitable for the truck platooning application. From the analysis of the simulator for the V2X communication utilities capabilities, it achieves the highest score among the simulators. It offers both default support for the large scale traffic simulation and also extension capabilities to provide more support. Therefore, if the project demands focus on large scale traffic simulation, the Gazebo simulator would be the effective simulator.

Thus, from the analysis of the simulators beginning with mapping the scenario steps to the KPIs and scoring them based on defined scoring parameters, it can be observed that the CARLA simulator has the highest overall score among the simulators. Consequently, the CARLA simulator is the effective simulator among the simulators in comparison for the truck platooning application. This result is arrived after detailed analysis of the simulator based on the selected scenario and analysing the capabilities of the simulation for the scenario. Moreover, the results can also be used to select simulators if the project demands are focused on a particular KPI which is also advantage of this method of evaluation of simulators.

# Chapter 6

# Conclusion and Future Work

The dissertation set out to find an effective simulator for the truck platooning application given a number of available simulators and simulators which is suitable for a middleware with ROS. Moreover, in this thesis, an approach was also proposed to evaluate the simulators to find an effective simulator. The main motivation behind this work was to effectively verify the truck platooning application using Hardware-In-Loop simulation testing methodology. Since the simulators play a major role in emulating the actual sensors and actuators in the HIL testing, the need for an effective simulator to perform the emulation is of high importance.

Initially, the background of the HIL Simulation testing and the importance of the simulation in the HIL testing was studied. This was followed with the understanding of the Scenario based testing and the existing TNO HIL testbed for which the simulators are evaluated. Subsequently, the truck platooning application was studied in detail to understand the demands of the application and to understand the scope of various scenarios that can happen in the truck platooning application. Consequently, the principal research question for this study was formulated which finds the effective simulator for the HIL simulation verification of the truck platooning application and an approach with which the effective simulator can be found.

This study was aimed at finding an effective simulator for truck platooning application given a number of available simulators that can perform the simulation. Therefore, the simulators that are to be compared and evaluated are chosen which are compatible with the HIL setup of TNO. This implied that the simulator should be capable of simulating autonomous vehicles, should be able to support scenario based testing and compatible with ROS environment. Therefore, the available simulators are studied and finally three simulators that are suitable for both the truck platooning application and that are compatible with the TNO HIL testbed are chosen. They are CARLA, LGSVL and Gazebo simulator which TNO currently uses for its HIL verification of Truck platooning.

Subsequently, an approach was required to find the effective simulator after selecting the simulators for comparison. The simulator choice and effectiveness can vary depending on the demands or the requirements of the application. Therefore, the first research question was aimed at finding the Key Performance Indicators of the simulators. A thorough literature study was conducted to identify the Key Performance Indicators for the simulators taking into account the truck platooning application requirements, the requirements of the TNO HIL environment and the generic requirements of an autonomous vehicle simulation. As a result of the literature study, a total of fifteen KPIs were formulated that are necessary for the simulator for the application under consideration.

Furthermore, the determination of the KPIs alone was not sufficient for the evaluation of the simulators. Since the number of KPIs are significantly high, the important KPIs among the selected KPIs needed to be determined. Consequently, the subsequent step in the proposed approach is the determination of the weight of the Key Performance Indicators. Therefore, the first sub-research question of RQ1 aimed at finding the weights of the KPI. To further simplify the complexity, the KPIs are segregated to Key Performance Areas with each KPI being assigned to

a specific Key Performance Area. To determine the weights of the KPIs, a multi-criteria decision making technique from the literature was used. Analytical Hierarchy Process is a multi-criteria decision making technique which aims to find the weights of the criterion in a multi-criteria domain. Based on this method a survey was conducted with the domain experts from diverse departments in TNO organisation to determine the weights of the KPIs. As a result of the survey, the Application Specific Aspects KPA had the highest weights. Therefore, the analysis and evaluation of the simulators were conducted based on the KPIs under the Application Specific Aspect KPA.

A method to validate the simulators based on the KPIs was needed to be determined. Therefore, the next step in the proposed approach is to determine a Use Case of the application based on which the simulators will be evaluated. Thus, the second research question aimed at determining the Use Case, the implementation of the Use Case in the three simulators and subsequently followed by the evaluation of the simulators based on the KPIs. This was answered by determining the Use Case which is Cutin Scenario in front of the lead vehicle in the platoon. This is one of the commonly occurring scenarios in the Truck platooning application and also this encompasses the KPIs that are under consideration. Subsequently, the scenario was implemented in the three simulators in which the platooning was done with Adaptive Cruise Control and not Cooperative Adaptive Cruise Control. Thus the implementation of the large scale traffic and Vehicle-Vehicle was not done in this thesis. However, the implementation methods and possibilities were thoroughly analysed for the complete implementation. The results of the Cutin were shown using a velocity profile plot. To facilitate the implementation and to create abstraction, the scenario was divided into scenario steps which are the common steps that needed to be implemented in all the simulators whereas the underlying implementation in each of the simulator would vary. Furthermore, the KPIs under consideration are mapped to the scenario steps. This proves the relevance of the selected scenario for the evaluation and also assists the evaluation. A scoring system was proposed based on three scoring parameters which helped in quantifying the analysis of the simulators based on the implementation details. The mapped KPIs are analysed and based on the analysis the simulators are assigned a score with respect to the three scoring parameters. As a result of the analysis, CARLA simulator proved to be effective in the Multi-agent Simulation and Large Scale Traffic simulation aspects. Gazebo simulator proved to be the suitable effective simulator for simulating Vehicle to Vehicle and Vehicle to Infrastructure communication aspects. LGSVL simulator proved to be a neutral choice based on the results. Overall, CARLA simulator achieved the highest results in this thesis and it is an effective simulator for truck platooning based on the result of this study.

Having answered the research questions, the main goal of this thesis has been achieved. Hence, it can be concluded that the thesis approach resulted in finding an effective simulator for the truck platooning application. However,it is not "the" effective simulator. There is no one simulator for all and thus this study provides a detailed analysis of the simulator capabilities which can be used to select simulator based on certain specific KPI demands.

Currently, the study only takes into account only the Application Specific Aspect KPIs from the complete list of the KPIs. Therefore, a future extension to this research is to take into account all the KPIs and determine different Use Cases to validate the other KPIs. Also, currently the implementation does not implement the large scale traffic and Cooperative Adaptive Cruise Control. The implementation of these aspects could provide further detailed analysis of the simulators. Finally, the proposed approach can be applied to the Hardware-In-Loop verification of any complex system and arrive at an effective simulator to perform the simulation. Therefore, a promising research direction is to extend this research to consider all the KPIs and to extend the study to any complex application.

# Bibliography


[1] HiL simulation in V-model. https://www.mathworks.com/help/physmod/simscape/ug/what-is-hardware-in-the-loop-simulation.html. Accessed: 27-11-2019.

[2] Dr Wältermann. Hardware-in-the-loop: The technology for testing electronic controls in automotive engineering, 01 2009.

[3] Components of HIL simulation. https://www.ni.com/nl-nl/innovations/white-papers/09/hardware-in-the-loop--hil--test-system-architectures.html. Accessed: 27-11-2019.

[4] Layered concept of truck platooning by ensemble. https://platooningensemble.eu/storage/uploads/documents/2019/02/11/ENSEMBLE-D2.2---Use-Cases-and-platooning-levels_disclaimer.pdf. Accessed: 09-01-2020.

[5] Till Menzel, Gerrit Bagschik, Leon Isensee, Andre Schomburg, and Markus Maurer. From functional to logical scenarios: Detailing a keyword-based scenario description for execution in a simulation environment, 05 2019.

[6] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator, Sep. 2004.

[7] Carla simulation maps and navigation. https://carla.readthedocs.io/en/latest/core_map/. Accessed: 20-07-2020.

[8] K. Bimbraw. Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology, July 2015.

[9] G. Marsden, M. Mcdonald, and Mark Brackstone. Towards an understanding of adaptive cruise control, 02 2001.

[10] J. Ploeg, B. T. M. Scheepers, E. van Nunen, N. van de Wouw, and H. Nijmeijer. Design and experimental evaluation of cooperative adaptive cruise control, Oct 2011.

[11] Jessica Cicchino. Effectiveness of forward collision warning and autonomous emergency braking systems in reducing front-to-rear crash rates, 02 2017.

[12] Accident mitigation by autonomous vehicles. https://www.rand.org/content/dam/rand/pubs/research_reports/RR400/RR443-2/RAND_RR443-2.pdf. Accessed: 20-11-2019.

[13] Safe autonomous vehicles. https://orfe.princeton.edu/~alaink/SmartDrivingCars/Papers/RAND_TestingAV_HowManyMiles.pdf. Accessed: 20-11-2019.

[14] E. Bringmann and A. Krämer. Model-based testing of automotive systems, April 2008.

[15] Lgsvl simulator overview. https://www.lgsvlsimulator.com/downloads/LGSVL-Simulator-Overview.pdf. Accessed: 21-11-2019.

[16] M. Schluse and J. Rossmann. From simulation to experimentable digital twins: Simulation-based development and operation of complex technical systems, Oct 2016.

[17] S. Ellwanger and E. Wohlfarth. Truck platooning application, June 2017.

[18] D. Jia, K. Lu, J. Wang, X. Zhang, and X. Shen. A survey on platoon-based vehicular cyber-physical systems. Master's thesis, Firstquarter 2016.

[19] Christophe Bonnet and Hans Fritz. Fuel consumption reduction in a platoon: Experimental results with two electronically coupled trucks at close spacing. Master's thesis, 08 2000.

[20] H. Hanselmann. Hardware-in-the-loop simulation testing and its integration into a cacsd toolset. Master's thesis, Sep. 1996.

[21] Maryam Kamali, Louise A. Dennis, Owen McAree, Michael Fisher, and Sandor M. Veres. Formal verification of autonomous vehicle platooning, 2017. Special issue on Automated Verification of Critical Systems (AVoCS 2015).

[22] J. C. Zegers, E. Semsar-Kazerooni, M. Fusco, and J. Ploeg. A multi-layer control approach to truck platooning: Platoon cohesion subject to dynamical limitations, June 2017.

[23] Johannes Glinz. A scenario-based approach to validating and testing software systems using statecharts, 08 2000.

[24] Ivar Jacobson, Magnus Christerson, Patrik Jonsson, and Gunnar Övergaard. Object-oriented software engineering - a use case driven approach, 1992.

[25] J. Banks, J. S. Carson, B. L. Nelson, and D. Nicol. Discrete-event system simulation, 2010.

[26] R. Sommet, D. Lopez, and R. Quere. From 3d thermal simulation of hbt devices to their thermal model integration into circuit simulators via ritz vectors reduction technique, May 2002.

[27] I. Kelander, A. N. Arslan, L. Hyvonen, and S. Kangasmaa. Modeling of 3d packages using em simulators, May 2004.

[28] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio López, and Vladlen Koltun. Carla: An open urban driving simulator, 2017.

[29] Robot operating system. `https://www.ros.org`. Accessed: 20-12-2019.

[30] Maryam Kamali, Louise A. Dennis, Owen McAree, Michael Fisher, and Sandor M. Veres. Formal verification of autonomous vehicle platooning, 2017. Special issue on Automated Verification of Critical Systems (AVoCS 2015).

[31] Formal verification of safety requirements of truc platooning. `https://pure.tue.nl/ws/portalfiles/portal/50926040/Tam_2016.pdf`. Accessed: 20-01-2020.

[32] F. M. Noori, D. Portugal, R. P. Rocha, and M. S. Couceiro. On 3d simulators for multi-robot systems in ros: Morse or gazebo?, Oct 2017.

[33] Tomasz Sulkowski, Paulina Bugiel, and Jacek Izydorczyk. In search of the ultimate autonomous driving simulator, 09 2018.

[34] Kenta Takaya, Toshinori Asai, Valeri Kroumov, and Florentin Smarandache. Simulation environment for mobile robots testing using ros and gazebo, 10 2016.

[35] M. C. Figueiredo, R. J. F. Rossetti, R. A. M. Braga, and L. P. Reis. An approach to simulate autonomous vehicles in urban traffic scenarios, Oct 2009.

[36] R. D. Gow, D. Renshaw, K. Findlater, L. Grant, S. J. McLeod, J. Hart, and R. L. Nicol. A comprehensive tool for modeling cmos image-sensor-noise performance, June 2007.

[37] Mohamed Eshtaiwi, Ibrahim Badi, Ali Abdulshahed, and Turan Erman Erkan. Determination of key performance indicators for measuring airport success: A case study in libya, 2018. JATM-Multiple Criteria Decision Making in Air Transport Management.

[38] Guide to airport performance measures. http://www.aci.aero/Media/aci/downloads/ACI_APM_Guidebook_2_2012.pdf. Accessed: 23-01-2020.

[39] Thomas L. Saaty. The analytic hierarchy process: Decision making in complex environments, 1984.

[40] Thomas Saaty. Decision making with the analytic hierarchy process, 01 2008.

[41] Thomas L. Saaty. The analytic hierarchy and analytic network processes for the measurement of intangible criteria and for decision-making, 2005.

[42] Carla simulation world. https://carla.readthedocs.io/en/latest/core_world/. Accessed: 20-07-2020.

[43] Lgsvl python api. https://www.lgsvlsimulator.com/docs/python-api/. Accessed: 21-07-2020.

[44] Peer-Olaf Siebers and Uwe Aickelin. Introduction to multi-agent simulation, 2008.

[45] Gazebo sensor class. https://osrf-distributions.s3.amazonaws.com/gazebo/api/dev/group__gazebo__sensors.html. Accessed: 26-07-2020.

[46] Gazebo wireless transmitter sensor. https://github.com/osrf/gazebo/blob/gazebo11/gazebo/sensors/WirelessTransmitter.cc. Accessed: 26-07-2020.

[47] Ros relay. http://wiki.ros.org/topic_tools/relay. Accessed: 26-07-2020.

[48] Carla ros bridge. https://carla.readthedocs.io/en/latest/ros_installation/. Accessed: 26-07-2020.

[49] Jason Sewall, David Wilkie, Ming Lin, and Pradeep Dubey. Interactive hybrid simulation of large-scale traffic, 08 2011.

[50] Ptv vissim simulator. https://www.ptvgroup.com/en/solutions/products/ptv-vissim/why-ptv-vissim/. Accessed: 26-07-2020.

[51] Carla traffic manager. https://carla.readthedocs.io/en/latest/adv_traffic_manager/. Accessed: 26-07-2020.

[52] M. Garzón and A. Spalanzani. An hybrid simulation tool for autonomous cars in very high traffic scenarios, 2018.

[53] Carla blueprint library. https://carla.readthedocs.io/en/latest/tuto_A_vehicle_modelling/. Accessed: 28-07-2020.

[54] Carla adding a vehicle. https://carla.readthedocs.io/en/latest/tuto_A_add_vehicle/. Accessed: 29-07-2020.

[55] Lgsvl vehicle dynamics. https://www.lgsvlsimulator.com/docs/ego-vehicle-dynamics/. Accessed: 30-07-2020.

[56] Lgsvl python api. https://www.lgsvlsimulator.com/docs/python-api/. Accessed: 30-07-2020.

[57] Lgsvl adding a vehicle. https://www.lgsvlsimulator.com/docs/add-new-ego-vehicle/. Accessed: 29-07-2020.

[58] Gazebo urdf format. http://gazebosim.org/tutorials?tut=ros_urdf. Accessed: 29-07-2020.