D4.3

Report on CCAM simulation tool landscape

Project short name
SUNRISE

Project full name
Safety assUraNce fRamework for connected, automated mobIlity SystEms

Horizon Research and Innovation Actions | Project No. 101069573
Call HORIZON-CL5-2021-D6-01

Funded by the European Union

ccam-sunrise-project.eu/

Dissemination level	Public (PU) - fully open
Work package	WP4: CCAM V&V framework
Deliverable number	D4.3: Report on CCAM simulation tool landscape
Deliverable responsible	Ilias Panagiotopoulos, ICCS
Status - Version	Final – V1.0
Submission date	22/07/2024
Keywords	simulation tooling specification, safety assurance framework, automated driving systems, verification and validation, use case requirements

Authors/Contributors

Name	Organisation
Ilias Panagiotopoulos	ICCS
Gerhard Weiß	VIF
Mauro Da Lio	UNITN
Dino Dodig	AVL
Jason Zhang	UoW
Jobst Beckmann, Philipp Legran	ika
Ashfaq Farooqui, Anders Thorsén	RISE
Georg Stettinger	IFAG
Tajinder Singh, Alperen Kiral	SISW
Gabriel Villalonga	CVC

Quality Control

	Name	Organisation	Date
Peer review 1	Olaf Op den Camp	TNO	18/06/2024
Peer review 2	Bernhard Hillbrand	VIF	15/06/2024
Peer review 3	Stefan de Vries	IDI	10/07/2024

Version history

Version	Date	Author	Summary of changes
0.1	23/10/2023	Ilias Panagiotopoulos (ICCS)	First draft of document structure
0.2	20/11/2023	Ilias Panagiotopoulos (ICCS)	Revised document structure
0.3	1705/2024	Mauro Da Lio (UNITN)	Edited UC1.2
0.4	20/05/2024	Mauro Da Lio (UNITN)	Edited CarMaker subsections in Section 2
0.5	22/05/2024	Mauro Da Lio (UNITN)	Edited text for UC3.1 and 3.2
0.6	11/06/2024	Ilias Panagiotopoulos (ICCS)	First version – ready for internal review
0.7	05/07/2024	Ilias Panagiotopoulos (ICCS)	Second version - Addressed comments from the internal review
0.8	15/07/2024	Ilias Panagiotopoulos (ICCS)	Third version - Addressed comments from the PC review
1.0	22/07/2024	Ilias Panagiotopoulos (ICCS)	Final version

Legal disclaimer

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Climate, Infrastructure and Environment Executive Agency (CINEA). Neither the European Union nor the granting authority can be held responsible for them.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS AND ACRONYMS

Abbreviation	Meaning
ACC	Adaptative Cruise Control
AD	Automated Driving
ADAS	Advanced Driving Assistance System
ADF	Automated Driving Function
ALKS	Automated Lane Keeping System
API	Application Programming Interface
ASAM	Association for Standarization of Automation and Measuring Systems
BSM	Basic Safety Message
C-ACC	Cooperativ-Adaptive Cruise Control
CAM	Cooperative Awareness Message
CCAM	Connected, Cooperative, and Automated Mobility
CPM	Cooperative Perception Message
C-V2X	Cellular-V2X
DDT	Dynamic Driving Task
DENM	Distributed Environmental Notification Message
DF	Driving Function
DSRC	Dedicated Short-Range Communication
ECU	Electronic Control Unit
ESC	Electronic Stability Control
ESP	Electronic Stability Programme (ESP)
FMI	Functional Mockup Interface

FMU	Functional Mock-up Unit
FOV	Field Of View
GLOSA	Green Light Optimised Speed Advisory
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GSR	General Safety Regulations
GT	Ground Truth
GUI	Graphical User Interface
HD	High Definition
HWP	HighWay Pilot
ICOS	Independent Co-Simulation
IDM	Intelligent Driver Model
IIS	Intersection Information Service
IMU	Inertial Measurement Unit
ISP	Image Signal Processing
IVI	In-Vehicle Information
KPI	Key Performance Indicator
LDM	Local Dynamic Map
MAP	MAp-related Message
MRM	Minimum Risk Maneuver
NEPCE	Nearly Energy-Preserving Coupling Element
OBU	On Board Unit
OD	Operational Domain
ODD	Operational Design Domain

OEDR	Object and Event Detection and Response
OEM	Original Equipment Manufacturer
OSI	Open Simulation Interface
PID	Proportional – Integral – Derivative
RoMPaC	Robust Motion Planning and Control
ROS	Robot Operating System
RSS	Received Signal Strength
RSU	Smart Road-Side Unit
SAF	Safety Assurance Framework
SOTIF	Safety Of The Intended Function
SPAT	Signal Phase & Timing
SR	Signal Request
SUMO	Simulation of Urban MObility
SuT	System Under Test
Tcl	Tool command language
TTC	Time to Collision
UC	Use Case
UDP	User Datagram Protocol
V&V	Verification and Validation
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything
VRU	Vulnerable Road User
VTP	Verification Test Procedure

WP	Work Package
XiL	X In the Loop (e.g., Software, Hardware, Model, Vehicle, etc.)

EXECUTIVE SUMMARY

Safety assurance of Connected, Cooperative and Automated Mobility (CCAM) technologies and systems is a crucial factor for their successful adoption in society, yet it remains to be a significant challenge. CCAM must be proven to be safe and reliable in every possible driving scenario within their Operational Domain (OD). It is generally acknowledged that for higher levels of automation, the validation of these systems by real-world driving tests would be infeasible mainly due to cost efficiency reasons. Instead, a mixture of virtual and physical testing is seen to be a promising approach, with the virtual testing bearing most weight. Furthermore, certification initiatives worldwide struggle to define a harmonized approach on safety assurance, which is hampering massive deployment of CCAM systems.

In the light of the above, the SUNRISE project aims to develop and demonstrate a commonly accepted, extensible Safety Assurance Framework (SAF) for the safety validation of a large variety of CCAM systems. The overall objective of the SUNRISE project is to accelerate the safe deployment of innovative CCAM technologies and systems for passengers and goods by creating demonstrable and positive impact towards safety, specifically to meet the EU's long-term goal of moving close to zero fatalities and serious injuries by 2050 (Vision Zero), and resilience of (road) transport systems. SUNRISE aims to achieve this, by creating and sharing a European federated database framework centralising detailed scenarios for testing of CCAM functions and systems in a multitude of relevant use cases, Also, a virtual harmonised simulation environment with standardised, open interfaces and quality-controlled data exchange will be established in SUNRISE project. SUNRISE will also work closely with CCAM stakeholders such as policy makers, regulators, consumer testing agencies, user associations and other relevant stakeholders.

One of the main goals of the SUNRISE project is to complete the targeted SUNRISE SAF by developing a harmonised verification and validation (V&V) simulation framework for CCAM systems in Work Package 4 (WP4). This deliverable presents the findings from task T4.3 which aimed to specify a tooling landscape, based on the definition of subsystems (T4.1) and the subsystem requirements mapping (T4.2), in order to virtually validate the CCAM functions and systems elaborated in the corresponding use cases on Work Package 7 (WP7). Each resulting tool should allow for the realistic simulation of the required subsystems, and provide adequate interfaces for data exchange with the federated Scenario Database (SCDB) developed in Work Package 6 (WP6).

The presented work is theoretical, so it may need to be updated or refined in further tasks and deliverables of the SUNRISE project. It focuses on pure virtual simulation where the needs and visions of all the partners in task T4.3 were considered to detail the specification of the simulation tools landscape, while supporting the subsystem requirements mapping performed in task T4.2. What could be stated here is that the specific tools or software presented in this deliverable, are NOT part of the SUNRISE SAF. The selected tooling landscape is only one of many options to virtually validate CCAM systems. In this context, a simulation tool selection

process is explained in this deliverable, which aims to be one of the main interests for SAF users.

1 INTRODUCTION

1.1 Project intro

Safety assurance of CCAM systems is crucial for their successful adoption in society, yet it remains a significant challenge. CCAM systems need to demonstrate reliability in their complete operational design domains (ODD), requiring robust safety argumentation. It is already acknowledged that for higher levels of automation, the validation of these systems by means of real test-drives is infeasible. In consequence, a carefully designed mixture of physical and virtual testing has emerged as a promising approach, with the virtual part bearing an increasingly significant weight in this mixture for cost efficiency reasons. Worldwide, several initiatives have started to develop test and assessment methods for automated driving functions (ADFs). These initiatives have already moved from conventional validation to a scenario-based approach and combine different test methods (physical and virtual) to avoid the million-mile issue.

The initiatives mentioned above provide new approaches to CCAM validation, and many expert groups formed by different stakeholders are already working on CCAM systems' testing and safety assurance. Nevertheless, in case a common European validation framework and homogeneity regarding validation procedures to ensure safety of these complex systems is found, the large-scale deployment of CCAM solutions will be accelerated. In this landscape, the role of standards is paramount in establishing common ground and providing technical guidance. However, standardising the whole pipeline of CCAM validation and assurance is in its infancy, as many of the standards are under development or have been very recently published and still need time to be synchronised and established as common practice.

Scenario databases are another issue tackled by several initiatives and projects. A federated approach should be used (at least at the European level) to enable the cooperation of many existing and potential new scenario database initiatives, and deal with scenarios of any possible variations, including the creation, editing, parameterisation, storing, exporting, importing, etc., in a universally agreed manner.

Furthermore, validation methods and testing procedures still lack appropriate safety assessment criteria to build a robust safety case. These shall be set and valid for covering the whole parameter space of scenarios. Another level of complexity is added, due to regional differences in traffic rules, signs, actors, and situations.

Evolving from the achievements obtained in predecessor project HEADSTART, and taking other initiatives as a baseline, it becomes necessary to move to the next level in the concrete specification and demonstration of a commonly accepted SAF for the safety validation of CCAM systems, including a broad portfolio of use cases and comprehensive test and validation tools. This will be done in SUNRISE (**S**afety ass**u**ra**n**ce f**r**amework for connected, automated mobil**i**ty **S**yst**e**ms).

1.2 SUNRISE project and the simulation framework

The Safety Assurance Framework is the main product of the SUNRISE project. This framework takes a central role, fulfilling the needs of different stakeholders that all have their own interests in using or applying it. The overall objective of the SUNRISE project is to accelerate the safe deployment of innovative CCAM technologies and systems for passengers and goods by creating and sharing a European federated Scenario Database Framework for testing CCAM functions and systems in a multitude of relevant test cases. The framework aims to the development and use of standardised, open interfaces and to enable quality-controlled data exchange.

Following the above, the SUNRISE harmonized V&V simulation framework for the virtual validation of CCAM systems developed in WP4 is a fundamental part of the SUNRISE SAF. Deliverable D4.1 [1] presented the findings from task T4.1, identifying relevant subsystems of the harmonised V&V simulation framework (see Figure 1), as described below:

- **Test case manager:** The “test case manager” subsystem’s main function is to orchestrate the execution of test scenarios in the simulation framework.

- **Environment:** The “environment” subsystem’s main function is to describe the surrounding environment in which the CCAM System under Test (SuT) operates.

- **Subject vehicle:** The “subject vehicle” subsystem includes the “Sensors”, the “AD function”, and the “Vehicle Dynamics”. The “Sensors” is a key element in enabling automated driving systems to provide both vehicle localisation and environmental perception of the vehicle's surroundings within its ODD [2]. The “AD function” controls the vehicle's actuators and maneuvers the vehicle. The “Vehicle Dynamics” describes the motion of a vehicle based on specific inputs (e.g., external, and internal forces).

- **Traffic agents:** The “traffic agents” subsystem simulates the behaviour of various types of dynamic elements except the subject vehicle (SuT).

- **Connectivity:** The “connectivity” subsystem enables communication between vehicles and other actors (pedestrians, cyclists, infrastructure elements in the surroundings).

- **Simulation model validation:** This subsystem is necessary to approve the quality and correlation to reality of a simulation model.

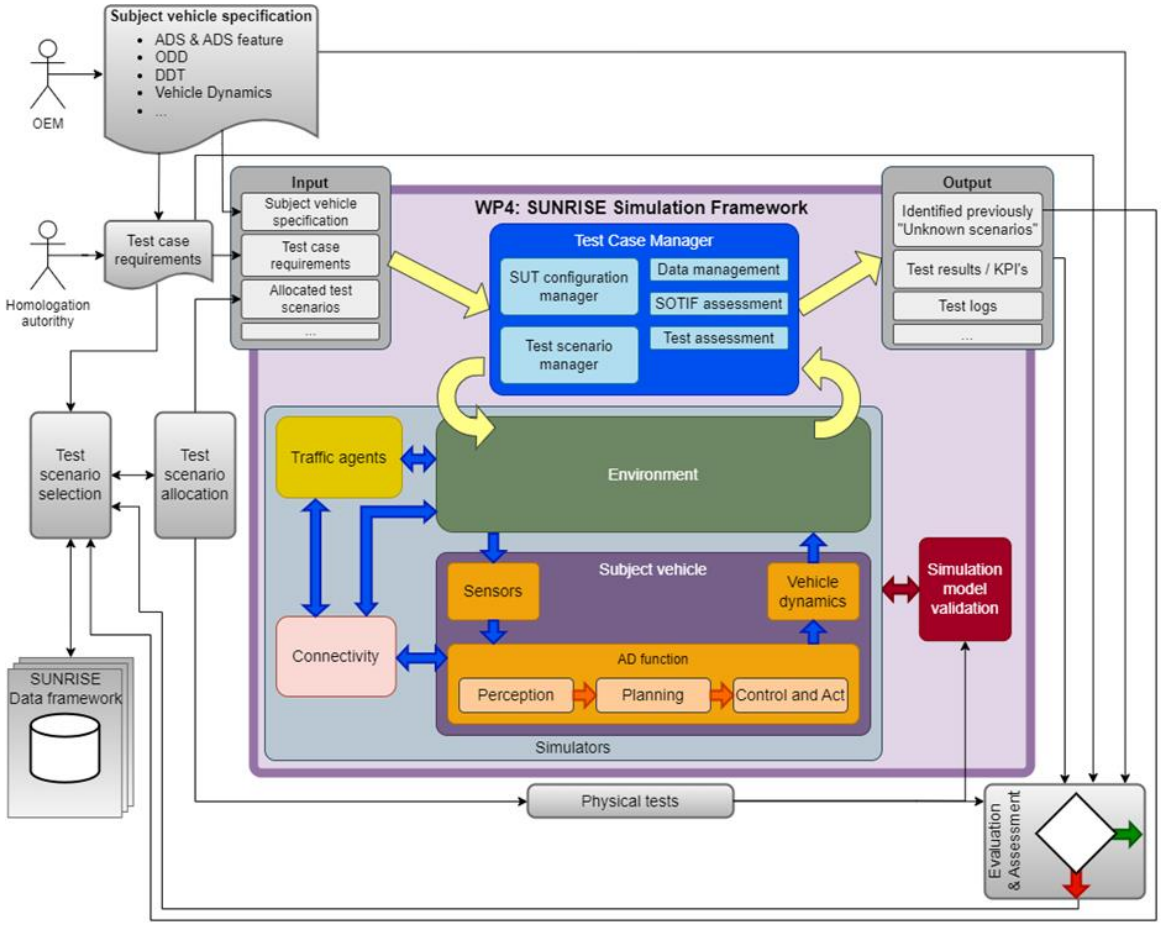

Figure 1: The proposed SUNRISE simulation framework with its subsystems [ref D4.1].

Task T4.2 had the goal to present the mapping between the requirements for all the use cases (T7.1) and the subsystems identified in task T4.1. Task T4.3 follows the work in T4.2, focusing on which requirements should be fulfilled by each subsystem of the SUNRISE simulation framework, and identifying a simulation tooling with the tools that best comply with all these requirements. More details about T4.1, T4.2 and T7.1 that are used as basis for task T4.3 can be seen in the deliverables D4.1, D4.2 [3] & D7.1 [4].

What could be stated here is that the specific tools or software presented in this deliverable, are NOT part of the SUNRISE SAF. The selected tooling landscape is only one of many options to virtually validate CCAM systems. In this context, a simulation tool selection process is explained in this deliverable, which aims to be one of the main interests for SAF users.

1.3 Purpose of the deliverable

As mentioned previously, in the SUNRISE project, WP4 aims to develop a harmonised V&V simulation framework for the virtual testing and validation of CCAM functions and systems, and completing the tooling with hybrid and real-world testing and validation approaches.

This deliverable presents the work done in the third task (T4.3) of WP4, where the involved partners have together identified and agreed on a non-exclusive list of simulation tools for the safety validation of the CCAM systems presented in the use cases elaborated in WP7.

More in detail, this deliverable has the following purposes:

1. **Selection of a tool set** that respects:
    A. The subsystem requirements defined in deliverable D4.1 section 4.
    B. The UC requirements assigned to each subsystem according to deliverable D4.2 (section 5).
    C. The total set of sub-UC requirements defined in deliverable D4.2 (section 8).
2. **Explanation of the tool selection process** applied in the form of guidelines.

Most of the partners that participated in task T4.3 have experience in simulation tools, with respect to the virtual validation of CCAM and ADAS systems, and the intention is that the definition of the simulation tooling specifications and the listed simulation tools shall be versatile and adoptable for future technology development. The partner contributions to this deliverable are summarized in Table 1.

Table 1: Partner contributions to D4.3.

Partner	Contributions
ICCS	Task leader and main editor for the deliverable D4.3. ICCS has also contributed to the virtual validation design of the sub-UCs 1.3 & 4.2, where multiple simulation tools co-exist.
AVL	Review and alignment with other partners on the tool necessary to cover the requirements for a simulation model. Created and presented Model.CONNECT to the partners as a co-simulation tool.
AVL AST	Review and alignment with other partners on the tool necessary to cover the requirements for a simulation model. Created and presented Model.CONNECT to the partners as a co-simulation tool.
AVL TR	Contribution by proposing virtual simulation tools within AVL-TR's field of expertise.
CAF	Participation in discussions about specifying a tooling landscape based on the CARLA simulator, sharing CAF's tools of choice for simulations, and ensuring the info-rmation, requirements, and context from Task T4.2 (from which we were Task Leader) was properly transmitted.
IFAG	Contribution by reviewing existing simulation tools for modelling the environment, including IFAG's ability to interface with radar sensor models as part of the overall simulation tool chain, including their parameterization capabilities. In particular, the environment simulation tool IPG CarMaker was analysed in terms of its ability to interface with the targeted high-fidelity radar models.

ika	Contribution by presenting simulation tools that are suitable to meet the requirements of T4.2 and that are currently used. These simulation tools were examined with particular interest in the level of detail of the simulations.
RISE	Bringing the subsystem requirements from sub-UC 4.1 to tooling landscape requirements.
RSA	Participation in discussions about the tooling landscape
SISW	Contributed with a review of existing tools and methodologies which meet the SOTIF requirements, a major part of the outlined general requirements. This is presented in Section 3.5. In addition, SISW gave an overview in Chapter 2 to SISW tools, Simcenter Prescan and HEEDS, and how they represent and fit w.r.t to the simulation framework subsystems. In Section 3.1.1, SISW contributed with a mapping of the requirements of sub-UC 1.1 to the two tools to understand how suitable they are for the usecase and what are the gaps.
UoW	Participation in discussions about the tooling landscape.
VICOM	Participation by presenting a virtual testing pipeline for perception.
VIF	Contribution by looking into tools for each defined subsystem that would fulfil the requirements of sub-UC 1.1.
CRF	CRF has left the project before providing content to this task.
CVC	Contribution focus on CARLA by collecting information on how we can use this virtual simulator for real applications; Contribution on listing which requirements CARLA can fulfil and its limitations regarding sub-UC 1.1, 1.2 and 1.3.
UNITN	Contribution to Section 2 in relation to CarMaker tools. Contribution to UC 1.2, 3.1 and 3.2 of Section 3.

1.4 Intended audience

The intended audience of the deliverable is primary the rest of WP4 but may also be relevant for the rest of the project consortium. The most important information for them can be found in the chapters 2 and 3.

Thus, the most relevant “consumers” of this deliverable are:

1. **Partners working on task T4.4**. They will have to develop a harmonized V&V Simulation Framework, based on the tool set selected in this deliverable.

2. **Partners working on tasks T7.2 and T7.3**. They need to be informed about the selected tool set, that they will have to apply in the execution of their tasks.

3. **External stakeholders**. For them, it is of high importance to understand the process applied for our tool selection, thereby enabling them to perform a similar process on their own. For example, V4SAFETY consortium [5] could be a target audience as it is not only develops a safety assessment framework by using virtual simulations for the assessment of a variety of safety measures, but also defines a simulation structure to guide users in setting up their simulations.

1.5 Structure of the deliverable and its relation with other work packages/deliverables

This deliverable is structured as follows: Chapter 2 gives an overview of the simulation tools for each one of the subsystems of the SUNRISE simulation framework, whereas chapter 3 addresses the coverage of subsystem requirements on tools, interfaces and fidelity. Chapter 4 presents a description of the simulation tooling specification according to the sub-UCs of WP7. Chapter 5 addresses critical points resulting from the gap analysis, which aims to find gaps by identifying simulation tool limitations and subsystems requirements, based on the results from Chapters 3 and 4. Chapter 6 summarises the conclusions.

Finally, two annexes are included; annex 1 contains the details of the CARLA OpenScenario 2.0 simulation tool features and improve the readability of the Chapter 2, and annex 2 presents tables with the additional requirements defined for sub-UCs 1.3 and 4.2.

2 REVIEW OF EXISTING TOOLS FOR THE VIRTUAL VALIDATION OF CCAM SYSTEMS

Simulation allows for virtual validation of CCAM systems in a huge number of scenarios, environments, and SuT configurations. A simulation uses models to represent the key characteristics and behaviours of the selected automated driving functions (ADFs) with various sensors, control elements, and different driving conditions over time. Of course, a simulation process does not make proving ground and real-world testis obsolete, but it can help focusing on the necessary proving ground and real-world tests to verify the simulation result. Additionally, the massive use of virtual simulation allows for significant cost savings and time reductions.

In the present chapter, a review of the existing simulation tools is provided with respect to the virtual validation of CCAM systems based on the main subsystems of the SUNRISE simulation framework. It is important to note here that the SUNRISE SAF is tool-agnostic. The SAF is designed to be flexible and adaptable, allowing project partners and future users to select the tools that best fit their specific needs and constraints. In this context, the aim of this chapter is not to prescribe a single set of tools, but rather to demonstrate a process of how suitable tools can be identified and mapped to the subsystem requirements.

The tool selection process applied in this chapter involves the following key steps:

- Identifying the relevant subsystems of the SUNRISE simulation framework and their requirements (D4.1 and D4.2).
- Reviewing the capabilities of existing simulation tools and their suitability for each subsystem.
- Mapping the subsystem requirements (D4.1 section 4 and D4.2 section 5) to the features of the selected tools, ensuring comprehensive coverage.
- Addressing any gaps or limitations of the tools.
- Validating the selected tooling against the total set of use case requirements (D4.2 section 8) and generic requirements (D4.2 section 4).

The tools selected in this chapter were chosen based on their ability to meet the subsystem requirements (D4.1 section 4) and the use case requirements assigned to those subsystems (D4.2 section 5).

2.1 Tools for the subsystem “test case manager”

In this section a review of the main existing simulation tools is provided which can support the “test case manager” subsystem.

2.1.1 Model.CONNECT™

Model.CONNECT™ is a model integration and co-simulation platform, connecting virtual and real components. Models can be integrated based on standardized interfaces (Functional Mockup Interface, FMI) as well as on specific interfaces to a wide range of well-known simulation tools. Additionally, **Model.CONNECT™** supports the organization of the system model variants. These variants may describe different configurations of the SuT as well as different testing scenarios and testing environments [6].

Through the integration with multiple model execution environments, **Model.CONNECT™** supports the continuous integration of functional scenarios that form the basis of model-based development processes in the automotive industry. Furthermore, **Model.CONNECT™** features model parametrization and batch simulation capabilities, simulation online monitoring, result analysis, and reporting functionalities. Interfaces to various optimization tools enable design studies and optimizations.

The model execution is supported in two flavors which can also be combined:

- Model integration based on models that are provided as executable libraries, i.e., FMI for Co-Simulation or Model Exchange. Such model configurations can be executed in one process. Such closely coupled model configurations are prepared for execution on real-time operating systems with later releases of **Model.CONNECT™**.

- Tool-coupling based on the Independent Co-Simulation (ICOS) technology which is a distributed co-simulation platform with a wide variety of supported simulation tools, co-simulation algorithms, i.e., adaptive time step control, Nearly Energy-Preserving Coupling (NEPCE), and the possibility to connect real-time systems to the co-simulation. The modular architecture provides the possibility for an iterative model developing process. Furthermore, influences of model accuracy, model depth, and nonlinearities on the result can be determined, due to the cross-domain considerations.

Figure 2 shows the Graphical User Interface (GUI) of the **Model.CONNECT™** to setup the co-simulation architecture and coupling settings.

Figure 3 shows the test case manager of the **Model.CONNECT™**. Parameters can be assigned to subsystems and for each simulation case different values (or paths) can set.

Figure 4 shows the results manager of the **Model.CONNECT™**, whereas input and output signals of each subsystem can be visualized. Moreover, a formula Calculator can be used to determine Key Performance Indicators (KPIs) or python script can be (automatically) executed to some further analysis.

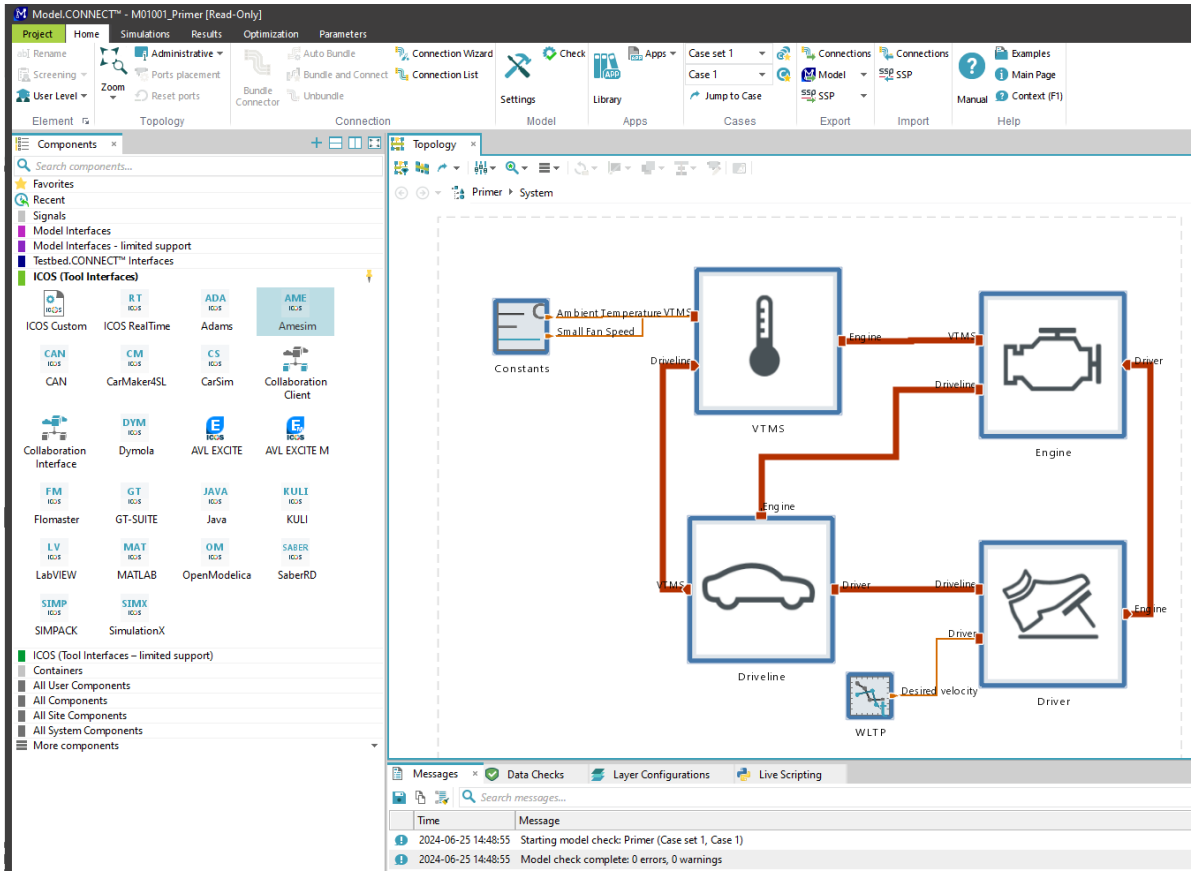

Figure 2: Model.CONNECT Graphical User Interface.

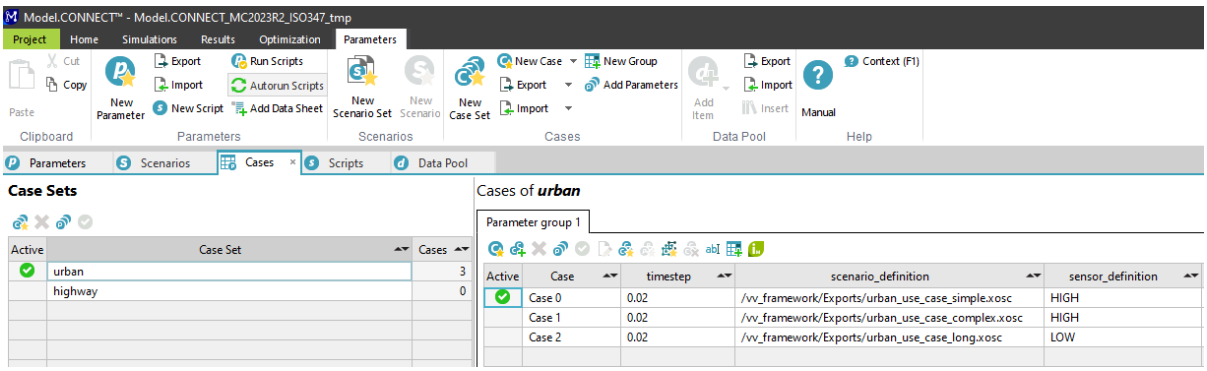

Figure 3: Model.CONNECT (Test) Case Manager.

2.1.2 CARLA

Within the **CARLA** ecosystem, the management of test cases is facilitated by a tool known as **ScenarioRunner** [7], primarily utilized in the **CARLA** leaderboard framework [8]. **ScenarioRunner** enables the testing of agents within pre-defined environments situated within already created maps, encompassing a collection of scenarios or cases that are triggered based on specific conditions achieved by the ego vehicle during simulation.

Scenario definitions within **ScenarioRunner** can be articulated using XML files. However, **ScenarioRunner** offers support for industry-standard formats such as OpenSCENARIO 1.0 [9] and OpenSCENARIO 2.0 (see annex 1), allowing for integration with external tools and frameworks.

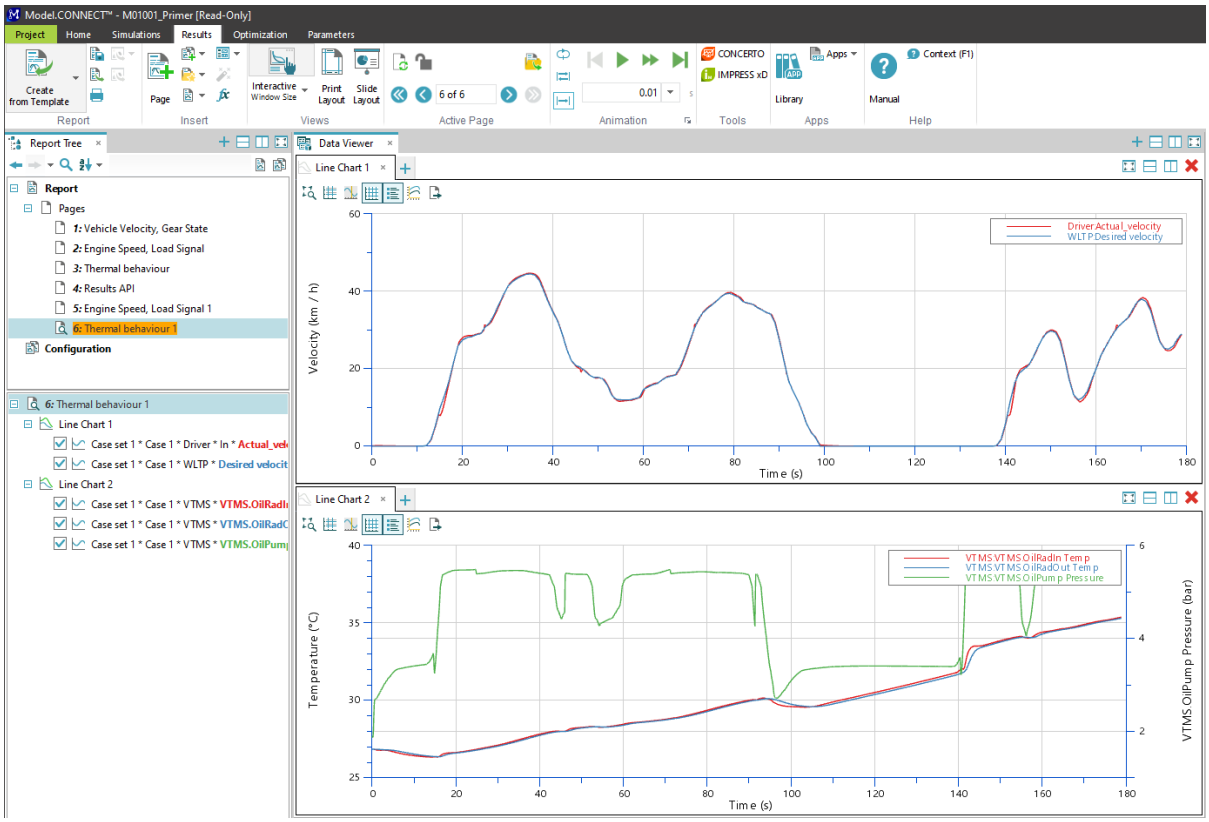

Figure 4: Model.CONNECT Results Manager.

Upon completion of a simulation run, **ScenarioRunner** generates a report detailing the evaluation of each scenario. This report provides insights into the performance of agents under various simulated conditions and scenarios. Some default conditions and metrics are already within the repository (e.g., collisions, running timeout, traffic laws infractions, distance to objects, etc.), whereas the repository allows the generation of new conditions and metrics through the **CARLA** python api [10].

2.1.3 Simcenter HEEDS and scene editor

Simcenter HEEDS [11] is a tool with functionality including process automation and orchestration, distributed execution, optimization studies, and data visualization. Figure 5 showcases the visualization of results in HEEDS from a design exploration study. In addition, the **scene editor** tool [12] enables the search space extraction for Safety Of The Intended Function (SOTIF) assessment for use in optimization studies. Thus, the two tools together are good prospects to cover the requirements outlined for the test case manager subsystem. However, the SuT configuration management is not covered within standard use of these tools and would require adaptation of the tools.

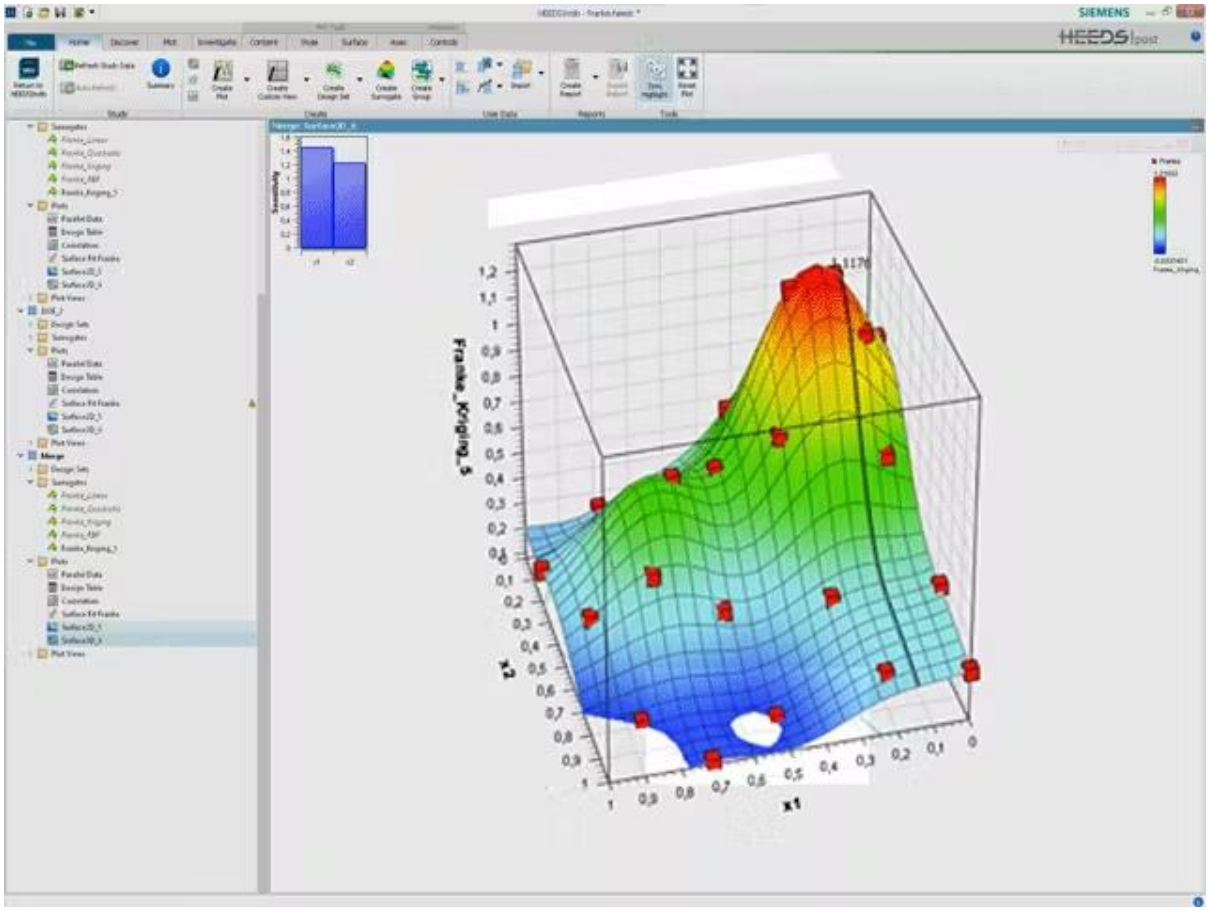

Figure 5: Screenshot from Simcenter HEEDS showing design exploration and visualization capabilities.

2.1.4 CarMaker

CarMaker is an industrial simulation tool for vehicle dynamics and and advanced driving automation functions. It allows virtual testing and embedded software development for the automotive industry. **CarMaker** allows modeling a great spectrum of road geometries and environmental conditions (surface conditions, wind, visibility, etc.). The subject vehicle model is a realistic multibody dynamics model with a large number of parameters. It can be customized to represent virtually any vehicle. A basic module to program traffic objects, including Vulnerable Road Users (VRUs), is available with which the behaviour of many agents can be programmed. The system includes basic and "physical" models of many types of sensors. The system also includes a "driver" agent which controls the ego vehicle. Finally, the system can be customized with ad hoc developed glue C++ (an Matlab/Simulink) functions that extends its capacity.

In this section a brief overview of **CarMaker** main functionalities is provided. Firstly, the GUIs that are present in **CarMaker** are introduced. Furthrmore, in the conclusion of this section, an example of the customized 3D animation is presented. The main GUI of the **CarMaker** is presented in Fig. 6. This interface is used to run single simulations. The interface allows to access all simulations settings such as the scenario editor, the vehicle editor, the manoeuvre details, simulation speed, variables logging, manoeuvre settings, etc.

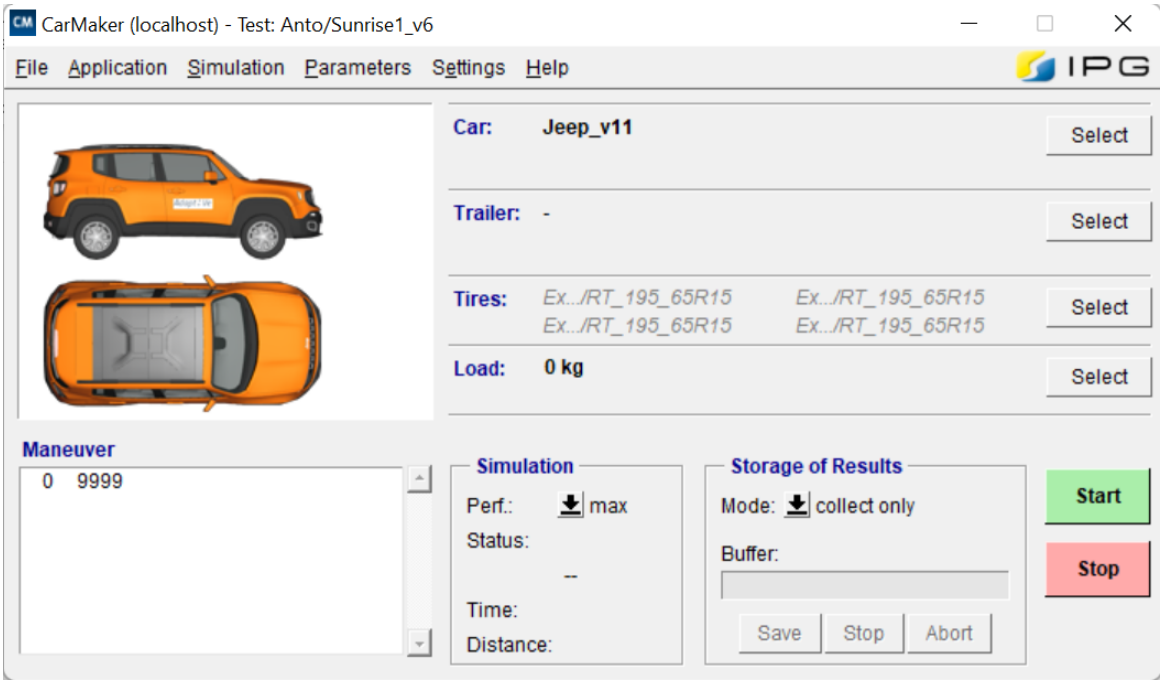

Figure 6: CarMaker Graphical User Interface.

Script control

Previously, GUIs that are involved in setting up running single simulations one by one were mentioned. In this section, the **Script Contro**l [13] terminal is described which is used in case of automatically executing multiple simulations in batches.

Script Control harnesses the inherent flexibility of the Tool command language (Tcl language) [14] to execute multiple simulation scenarios. Tcl is a bash-like programming language that can be used to call high level **CarMaker** and bash functionalities such as “load this scenario”, “change this parameter”, “execute this simulation”, “save this file in this folder” and so on. With this capability, it is possible to automate a range of tasks, such as: running a unit test program that evaluates the reliability of the agent at each release or running batches of simulations to extract results in the cases of a test matrix. Also, complex tasks can be created with **Script Control**. In Fig. 7 an example of the **Script Control** terminal is depicted, where 1000 simulations are executed with randomized parameters.

In the following basic example, written here as pseudocode, the simulation scenario called “Sunrise1_v4” is run 10 times in a for loop, each time with a different random value of the start position of the ego vehicle. At the end of each run the final velocity of Ego is displayed:

```
LoadTestRun "batches/Sunrise1_v4"
for {k = 0} {k < 10} {k++} {
        EGO_startPosition = random value between 30 and 60m;
StartSimulation;  WaitForSimulationEnd;
Log “Final Ego velocity = $Car.v m/s”;
}
```

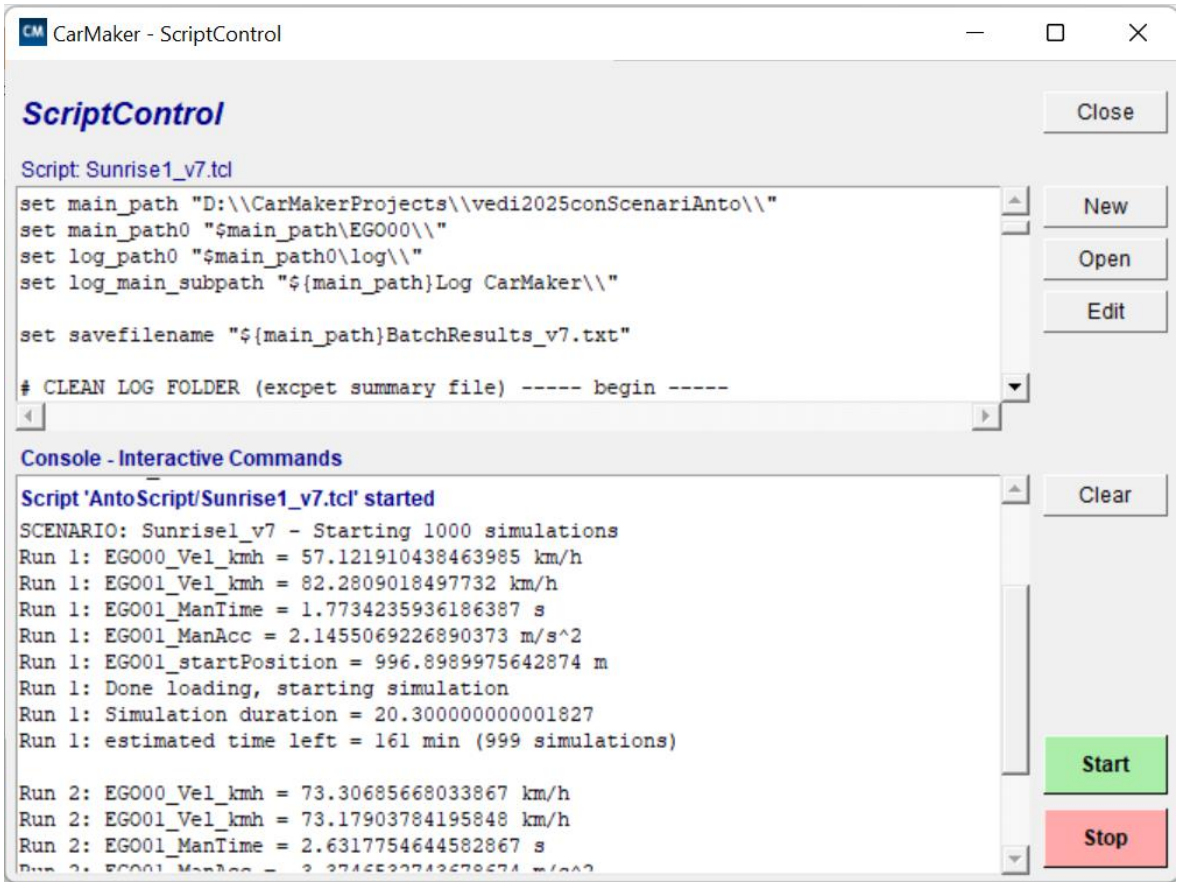

Figure 7: CarMaker Script Control terminal: Tlc code is read from the file at the top and the output is displayed in the terminal at the bottom.

Data analytics

Data analytics include a set of tools for the analysis of large simulation sets implemented in the Wolfram Language. Among these, there is a Machine Learning model to construct a stochastic function approximation of the vehicle performance over the logical scenario, i.e., a function that predicts the vehicle performance for any point within ranges of the scenario parameters. The function is made of a neural network with appropriate architecture and hyper parameters. It is trained with a limited number of simulations (typically 5000 simulations for a 5-dimensional parameter space). The function predicts collision, near miss, and safe states. Training is carried out with the bootstrapping method, and hence the trained model predicts the probability of one event and, also, the uncertainty or confidence level.

The trained model is a synthetic model of the vehicle performance, with known confidence, over the logical scenario. It can be used for different evaluations:

- To predict the safety of a vehicle (risk of collision) for a large set of given plausible events within the logical scenario (e.g., from naturalistic driving data).

- To derive aggregate metrics with known confidence for the vehicle performance, without carrying out many individual simulations [15]. It should be stated that with high-dimensional parameter spaces, the number of simulations that may be required for probing the space grows exponentially, i.e., a 5-dimensional space with 10 different values for each parameter would need 100000 different simulations.

- To identify failure modes (i.e., to identify clusters/regions in the logical scenario that fail for the same reason).

- To compare different agents (or versions of the same agent) on different regions of the logical scenario.

In addition to the tools above, **CarMaker** itself has several reporting tools. For example, it is possible to inspect any internal signal (see Fig. 8), display animations (see Fig. 9) or even realize custom animated views (see Fig. 10). More in detail, in Fig. 10, the top text lines show some key configuration parameters, whereas the bottom charts show important real time information that are helpful to understand the underlying decision making of the agent.

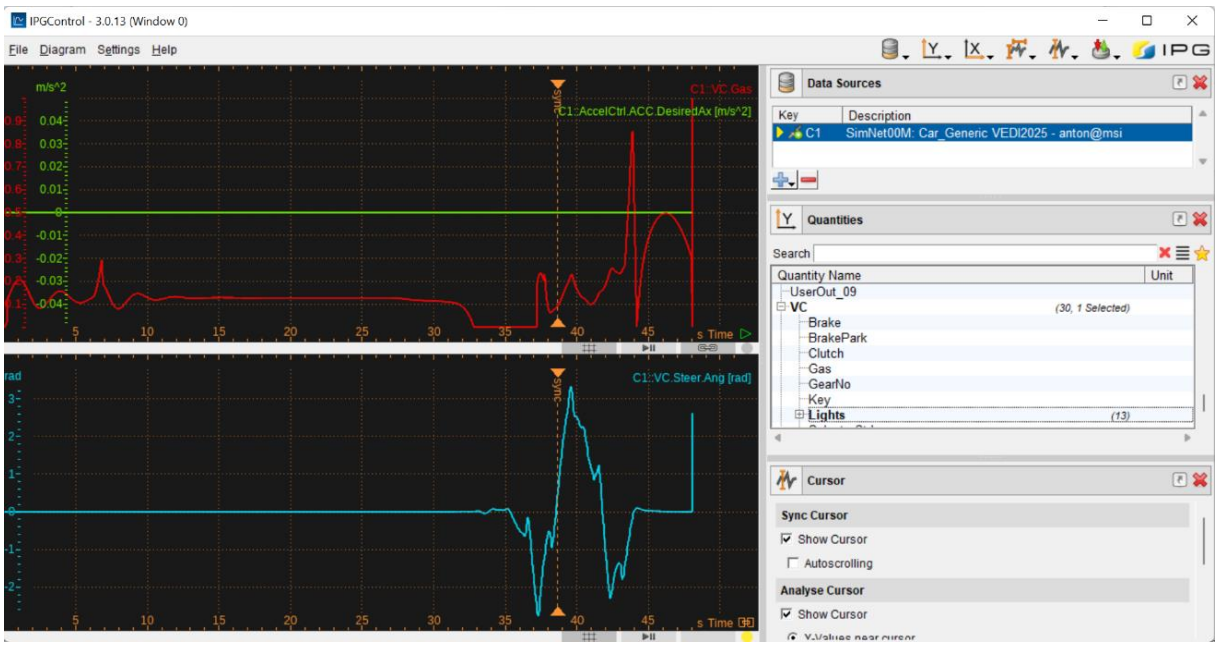

Figure 8: CarMaker signal plotting interface.

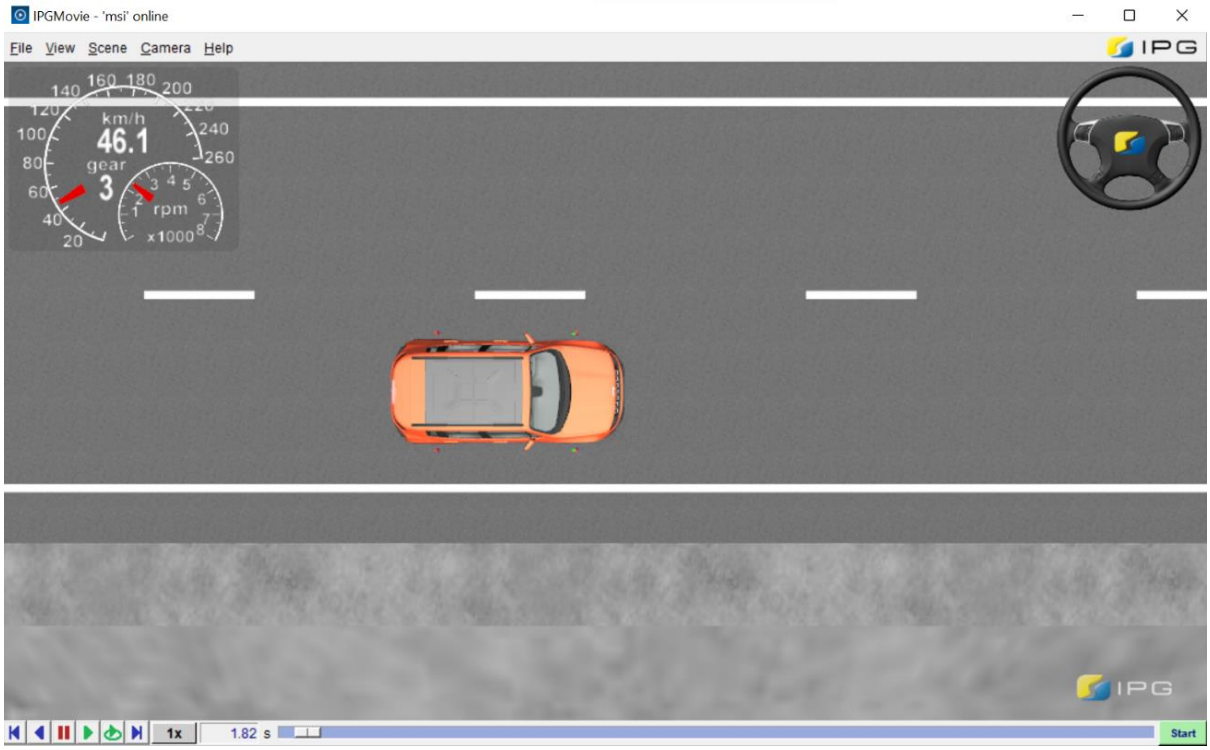

Figure 9: CarMaker 3D animation interface (default by IPG).

Figure 10: CarMaker 3D animation interface with customized OpenGL Codriver objects.

2.2 Tools for the subsystem “environment”

In this section a review of the main existing simulation tools is provided which can support the “environment” subsystem.

2.2.1 Esmini

Esmini is a software tool to play OpenSCENARIO files. It is provided both as a stand-alone application and as a shared library for linking with custom applications (see Fig. 11). In addition, some tools have been developed to support design and analysis of traffic scenarios [16].

It contains the following main libraries:

- **RoadManager (esminiRMLib)** - A library providing an interface to road networks described in the OpenDRIVE format.

- **ScenarioEngine (esminiLib)** - The main library providing a viewer and Application Programming Interface (API) to traffic scenarios described in the OpenSCENARIO format. This library includes RoadManager.

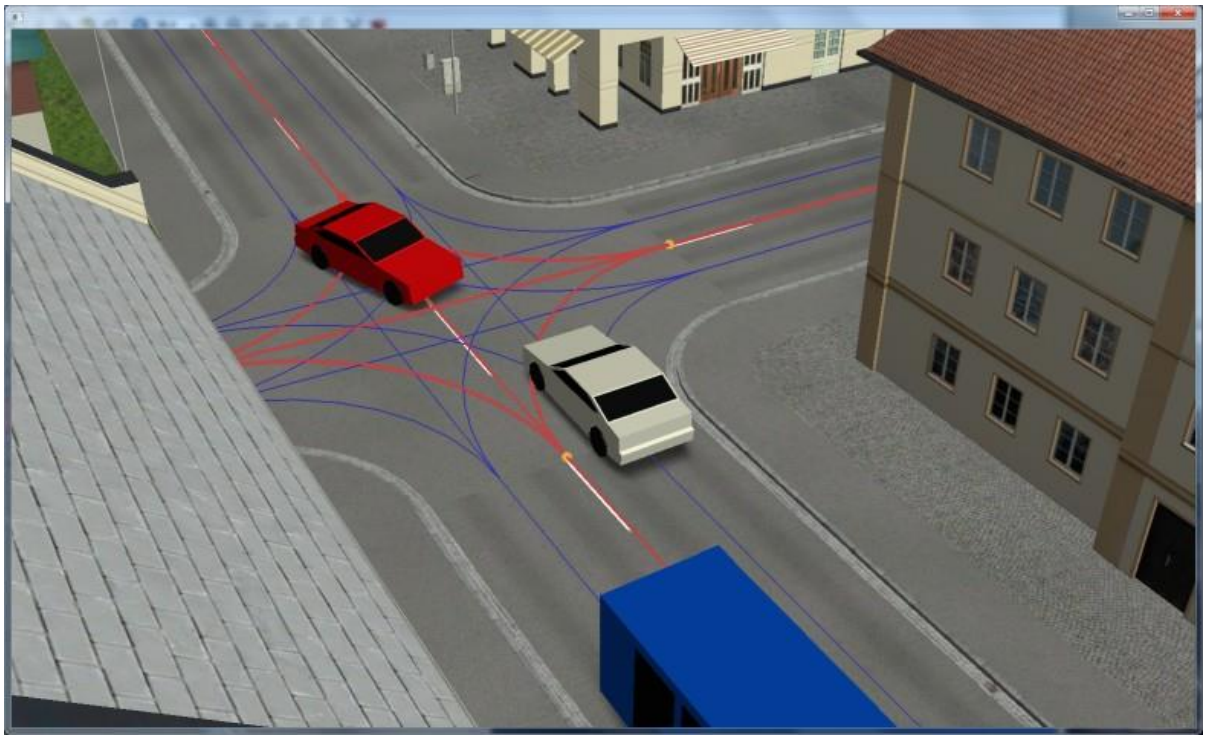

Figure 11: Esmini simulation tool.

It also contains a few applications that can be used as is or provide ideas for customized solutions:

- **esmini.** A scenario player application linking esmini modules statically.

- **esmini-dyn.** A minimalistic example using the esminiLib to play OpenSCENARIO files.

- **odrplot.** Produces a data file from OpenDRIVE for plotting the road network in Python.
- **odrviewer.** Visualize OpenDRIVE road network with populated dummy traffic.
- **replayer.** Re-play previously executed scenarios.
- **osireceiver.** A simple application receiving OSI messages from esmini over User Datagram Protocol (UDP).

The tool **Esmini** supports OpenSCENARIO v1.1 and v1.0. In order to run older OpenSCENARIO versions (i.e. v0.9.1), the Association for Standarization of Automation and Measuring Systems (ASAM) provides a transformation scheme that can be used with tools for automatic migration of XML files.

Additionally, **SUMO** is a open-source traffic simulation tool. It has been integrated with **Esmini**, making it possible to run **SUMO** simulations in **Esmini**, and even mix or co-simulate **SUMO** vehicles with OpenSCENARIO vehicles [17-18].

2.2.2 CARLA

In the realm of research and development, the definition of test environment within the **CARLA** simulator varies from basic setups to highly complex scenarios. Some samples of the complex scenarios can be found in the **CARLA** repository in the form of maps and are publicly available for all users. These maps offer a comprehensive 3D representation enriched with various elements for realistic simulation, including junctions, roads, traffic lights, signals, and more. **CARLA** maps, outlined in the documentation [19], provide a framework for modifying the existing maps that mimic real-world driving conditions.

Conversely, in case that the users need to replicate real road structures, the test environments can be defined using OpenDRIVE files, which provide a skeletal representation of a 3D scene featuring predefined road layouts. This method, detailed in the **CARLA** documentation [20], generates a barebones 3D environment with roads specified in the OpenDRIVE format. **CARLA** works with OpenDRIVE 1.4 standard.

Moreover, the Python API in **CARLA** enables dynamic modification of certain aspects of the map during simulation runtime. This functionality allows the users to introduce dynamic elements such as additional traffic (comprising vehicles and pedestrians), manage traffic lights dynamically, and manipulate environmental conditions such as weather and sun position based on the time of day. By leveraging the Python API's capabilities, users can create versatile and adaptive test environments that simulate a wide range of driving scenarios with varying levels of complexity and realism.

2.2.3 Simcenter Prescan

Simcenter Prescan is a simulation platform specifically designed for the development and testing of advanced driver assistance systems (ADAS) and autonomous vehicles. It allows engineers to simulate real-world scenarios and environments, such as urban traffic, highways, and various weather conditions, to evaluate the performance and safety of these systems [20].

Prescan provides a realistic virtual environment where engineers can model vehicles, surroundings, buildings, traffic lights, bridges, roads, etc. Additionally, any 3D model that can be imported into the 3D modelling software is supports, i.e., Blender [21] can be exported as a Prescan type 3D object.

It also enables users to make use of standardized interfaces to create a test scenario. As an example, user can import OpenDRIVE and OpenSCENARIO files to build road infrastructure and define test case scenarios.

On top of that, the software enables changing weather (fog, rain, and snow with various intensities) and lightning conditions (sun's angle or brightness of the scene) in the simulation, which has direct effect on the vehicle's sensors (see Fig. 12).

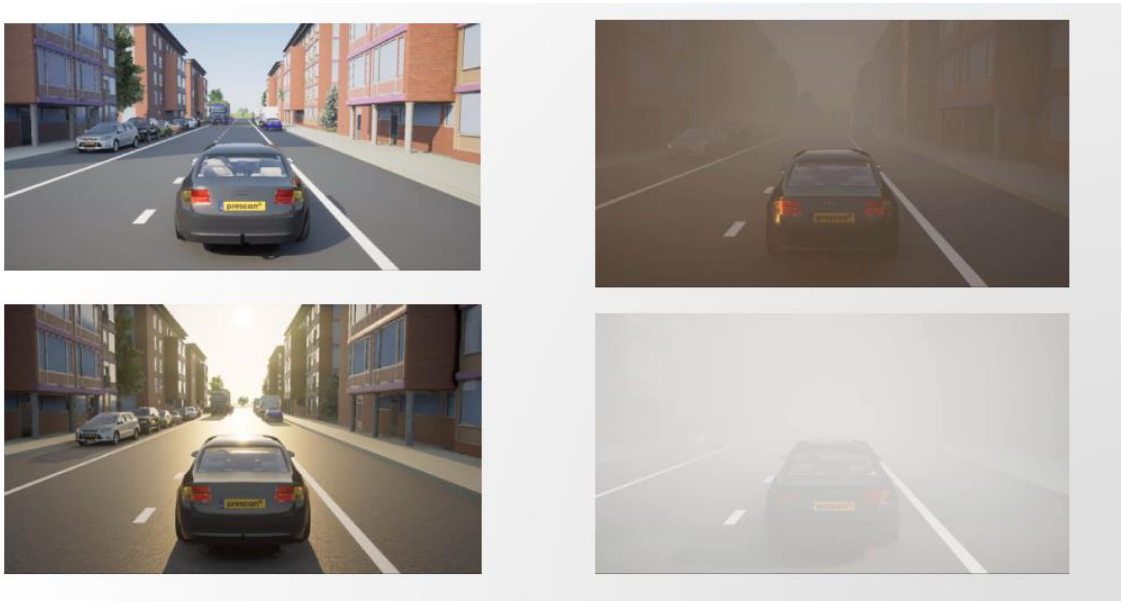

Figure 12: Photo-realistic environment conditions simulation in Simcenter Prescan.

2.2.4 CarMaker

The **CarMaker** scenario editor [22] is shown in Fig. 13. In this example two cooperative vehicles cross at a roundabout. Each vehicle has its own predefined path (orange/red lines).

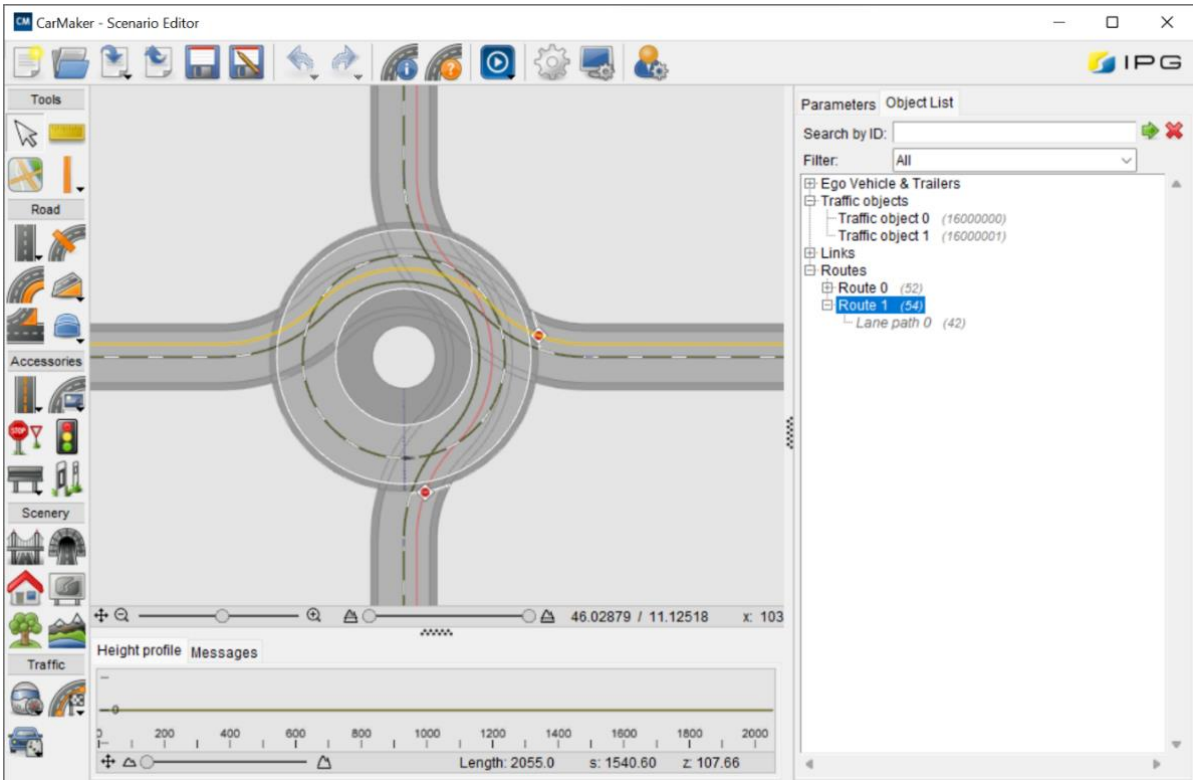

Figure 13: CarMaker Scenario editor.

The hierarchical structure of road entities that is shown in Figures 14 and 15 allows great flexibility when building a road.

```
road
    |->junction0
    |->junction1
    |->link0
    |->link1
    |       |->node0
    |       |->node1
    |       |->lanesection0
    |       |->lanesection1
    |       |       |->laneleft0
    |       |       |->laneleft1
    |       |       |->laneright0
    |       |       |->laneright1
    |->route
```

Figure 14: Road entities (picture from [22]).

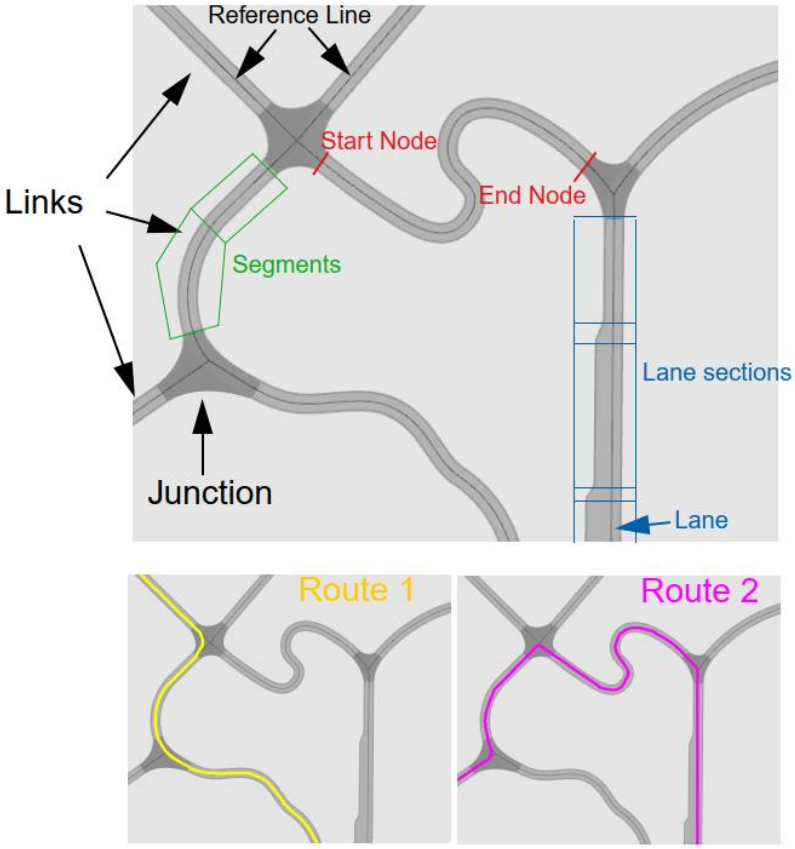

Figure 15: Road routes (picture from [22]).

Additionally, Fig. 16 shows some examples of road geometries. Elevation slope and camber can be assigned to roads. It is possible to customize the scenarios by means of a number of features, either graphical (road marking, road signs, etc.) or functional/environmental (speed limits, stop signals, environmental wind etc.), as shown in Fig. 17.

Finally, the **CarMaker** Scenario Editor allows to read/import the following formats of road/scenarios:

- KML files (WGS84-coordinates)
- ASCII files (cartesian coordinates)
- CRG-files
- OpenDRIVE®

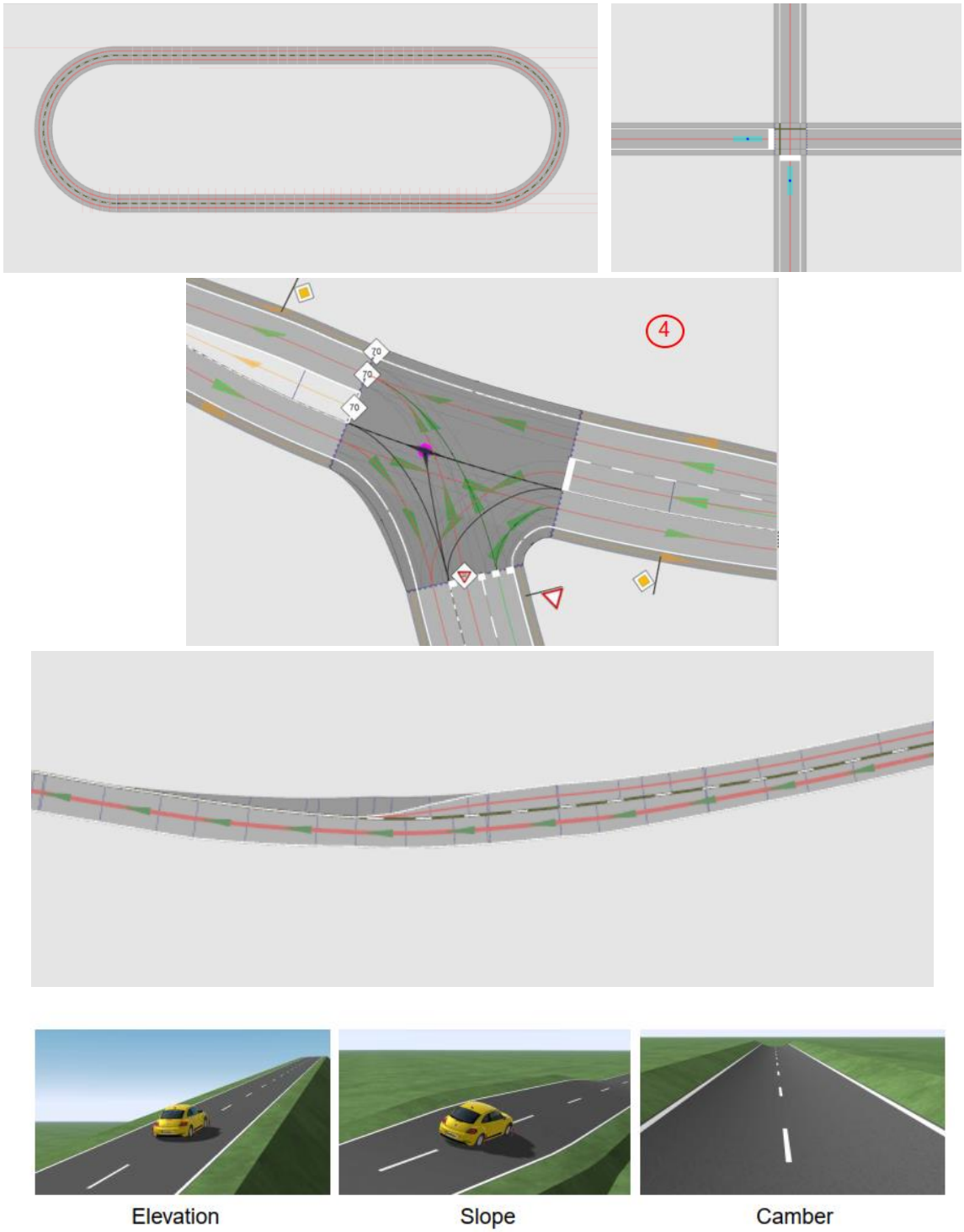

Figure 16: Examples of road geometries: ring, intersection, junction,, multiple lanes with variable width, elevation, slope, camber.

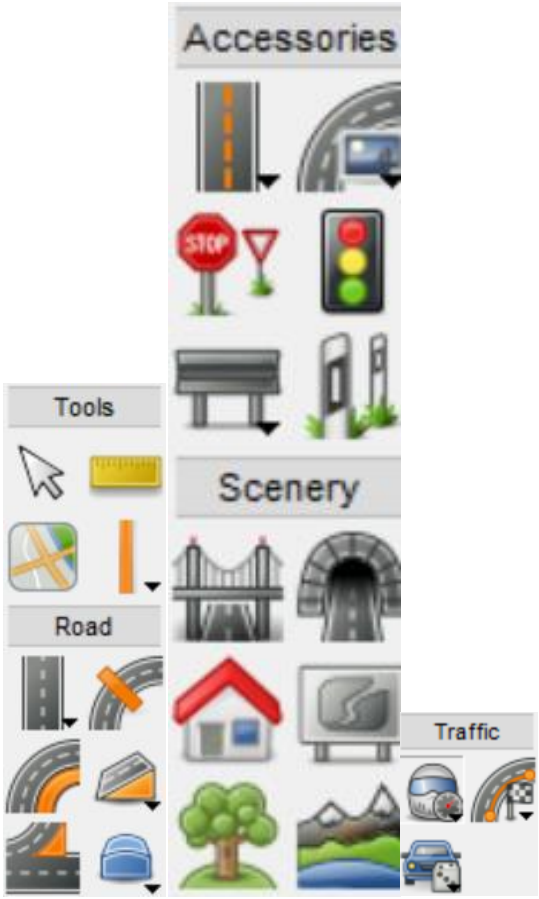

Figure 17: Features of the CarMaker Scenario Editor.

2.3 Tools for the subsystem “subject vehicle”

In this section a review of the main existing simulation tools is provided which can support the “subject vehicle” subsystem.

2.3.1 Simcenter Prescan

Sensor Models

There are five different kinds of sensor models in **Prescan**:

- Ground Truth Sensors: These sensors output ground truth sensor data and thus providing reference data against which other sensors or algorithms can be compared or calibrated.

- Idealized Sensors: They represent the group of sensors that are fast but less accurate. Instead of observing the exact geometry, they have bounding-box based detections.

- Detailed Sensors: This group represent a set of specialized sensors modelling that have average accuracy and reasonable execution times.

- Probabilistic Sensors: The models allow for more realistic simulation of radar and camera sensors through injection of typical sensor errors into the simulated object lists.

- Physics-based Sensors: The most realistic sensor modelling group in **Prescan** that enables realistic sensor outputs, which are costly in computer resources as well.

Depending on the desired level of fidelity in the simulation model, the user has different sensor options to choose from **Prescan**. Commonly used types include camera, lidar, radar, and ultrasonic sensor models.

AD Function
Even though **Prescan** does not natively have a default ADF, it enables the user to implement their algorithms in C++ or Simulink environment. Sensor outputs from **Prescan** can be reached in these environments and actor's updated states can be sent back to **Prescan** as output.

Vehicle Dynamics
Prescan natively supports multiple options for vehicle dynamics:

- 2D Simple: A bicycle model that is capable of simulating a car’s longitudinal, lateral and roll motion.

- 3D Simple: This model is capable of simulating a car's longitudinal, lateral, vertical, pitch and roll motion including modelling of suspensions.

- Amesim Dynamics: A 15 DOF vehicle model is supported which can be configured based on vehicle geometry & vehicle category (segment A, segment C, segment EV, SUV & truck).

- User-Specified Model: Simulink based vehicle dynamics model specified by the user, either via Simulink model or functional mock-up unit (FMU).

2.3.2 CARLA

Vehicle Models
The **CARLA** vehicle management API facilitates integration of a diverse array of vehicles from the official **CARLA** catalogue [23] into simulations. Users can incorporate vehicles with varying characteristics, such as size, speed, and handling. Beyond mere integration, the **CARLA** API enables users to manipulate the appearance of vehicles and control specific elements such as lights, doors, and other customizable features.

Sensor Models

CARLA uses a diverse variety of sensor types [24] tailored to various aspects of perception, ground truth estimation, and vehicle-state monitoring:

- Perception sensors: Sensors responsible for perceiving the environment, including Radar for detecting objects and velocities, RGB cameras for visual information, LIDAR for detailed 3D representations, inertial measurement unit (IMU) for motion tracking, Global Navigation Satellite System (GNSS) for location determination, and Received Signal Strength (RSS) for roadside environmental data.

- Ground truth sensors: Sensors providing detailed environmental understanding, such as semantic segmentation for pixel-level object classification, instance segmentation for delineating individual objects, semantic LIDAR segmentation for 3D environment understanding, obstacle detection for hazard identification, depth cameras for depth perception, optical flow for motion estimation, and lane invasion detection for monitoring lane integrity.

- Vehicle sensors: Sensors used to get the interaction of the agent with other agents. It only includes a collision sensor to register each time that the agent has a collision per frame, including collision with static objects, such as buildings and bushes.

In terms of sensor fidelity, **CARLA** has compatibility for 3d-party software to define the sensors used:

- Unreal sensors: Default sensors in **CARLA**, offering simplified renderings of real-world sensors and scenarios within the Unreal Engine environment. Camera parameters are not the same as real-world cameras.

- NVIDIA Omniverse sensors [25]: Engineered to emulate real-world cameras, striving for high fidelity and accuracy in replicating sensor behaviour and properties.

AD function

CARLA allows the assignment of the predefined traffic agents behaviours to ego vehicles [26], though users can also designate unique agents for each ego vehicle. Custom agents are created using the **CARLA** Python API, connecting vehicle commands via Ackermann steering, acceleration, and braking commands.

Vehicle Dynamics

CARLA's API offers customization options for defining the physics behaviour of vehicles within simulations. Users can tailor parameters related to the gears system, wheel physics, and other dynamic aspects of vehicle behaviour to simulate driving conditions. Moreover, **CARLA** provides the flexibility to switch between different physics subsystems, including PyshX [25] and Chrono [26], allowing users to choose the most suitable physics engine for their specific needs. It is worth highlighting that, when using Chrono, collisions are not supported, so **CARLA** will revert to its default physics.

2.3.3 WayWise

WayWise [27-28] is a rapid prototyping library developed for connected and autonomous vehicles. It is designed to explore the use of autonomous vehicles such as cars, tractors, and drones, with a focus on functional safety and cybersecurity.

The library facilitates the development of vehicle on-board systems and desktop control applications, enabling communication through the MAVLINK protocol. While not intended for production, **WayWise** allows for the exploration of specific use cases through rapid prototyping. It supports various vehicle types and offers sensor integration, including GNSS, IMUs, and Ultra Wide Band. Additionally, it integrates DepthAI for prototyping AI-enabled functionalities, such as maintaining a safe distance from or following persons.

2.3.4 VSM

VSM™ delivers simulations of complete virtual vehicle prototypes of passenger cars, trucks, tractors, and more – from the early concept and design phase to the validation and acceptance phase:

- Vehicle dynamics
- Flexible for different vehicle types
- Various existing track templates or use of your own data from driven routes, Global Positioning System (GPS), or Google Maps for realistic driving experiences

Sensor Models
VSM does not have any direct ADAS/AD related sensor models. Furthermore, it does not support standalone ADAS sensor simulation. **VSM** is intended as highly specialized advanced vehicle dynamics software with high flexibility of co-simulation with different tools or via FMU. **VSM** offers smooth interface with **Model.CONNECT** to make use of its ADAS/AD potential.

VSM is often used for creating detailed vehicle digital twin model with 3D realistic sensor mounting and vehicle geometry that supports various kinds of roads (paved road, off-road, hill, etc.). Appropriate to use for different types of impacts of roads on sensors, like vehicle shaking, pitching when braking, hard cornering tilting and other impacting factors.

AD Function
VSM does not have any option for a standalone ADF simulation, but it is used to enable ADF testing by predicting advanced vehicle dynamics phenomena (e.g. vehicle pitching in case of hard braking and its impact on ADAS/AD function, perception, fusion, etc.).

Vehicle Dynamics
VSM offers multiple options for vehicle dynamics, from very simplified models to highly complex multi-axle systems in different market segments (see Fig. 18).

VSM additionally offers multiple control algorithms for different strategies, road types or road surfaces with different traction models including tire models (see Fig. 19).

Real-time simulation enables integration with Hardware in the Loop (HiL) and Hardware in the Loop (SiL) environments, which is an essential part of Electronic Control Unit (ECU) testing.

Multibody dynamics model the vehicle mechanical systems like suspension, steering and chassis, and enables in-depth analysis of vehicle handling and stability.

Figure 18: AVL VSM Market Segments Overview.

Figure 19: AVL VSM Different tracks with different traction and dynamics.

2.3.5 CarMaker

CarMaker is an industry-focused simulation software designed for comprehensive development and testing of vehicles at all stages (SiL, MiL, HiL, ViL). The **CarMaker** vehicle Editor is shown in Fig. 20. Each tab on the left contains several classes of configuration parameters that allow the description of various vehicle submodels.

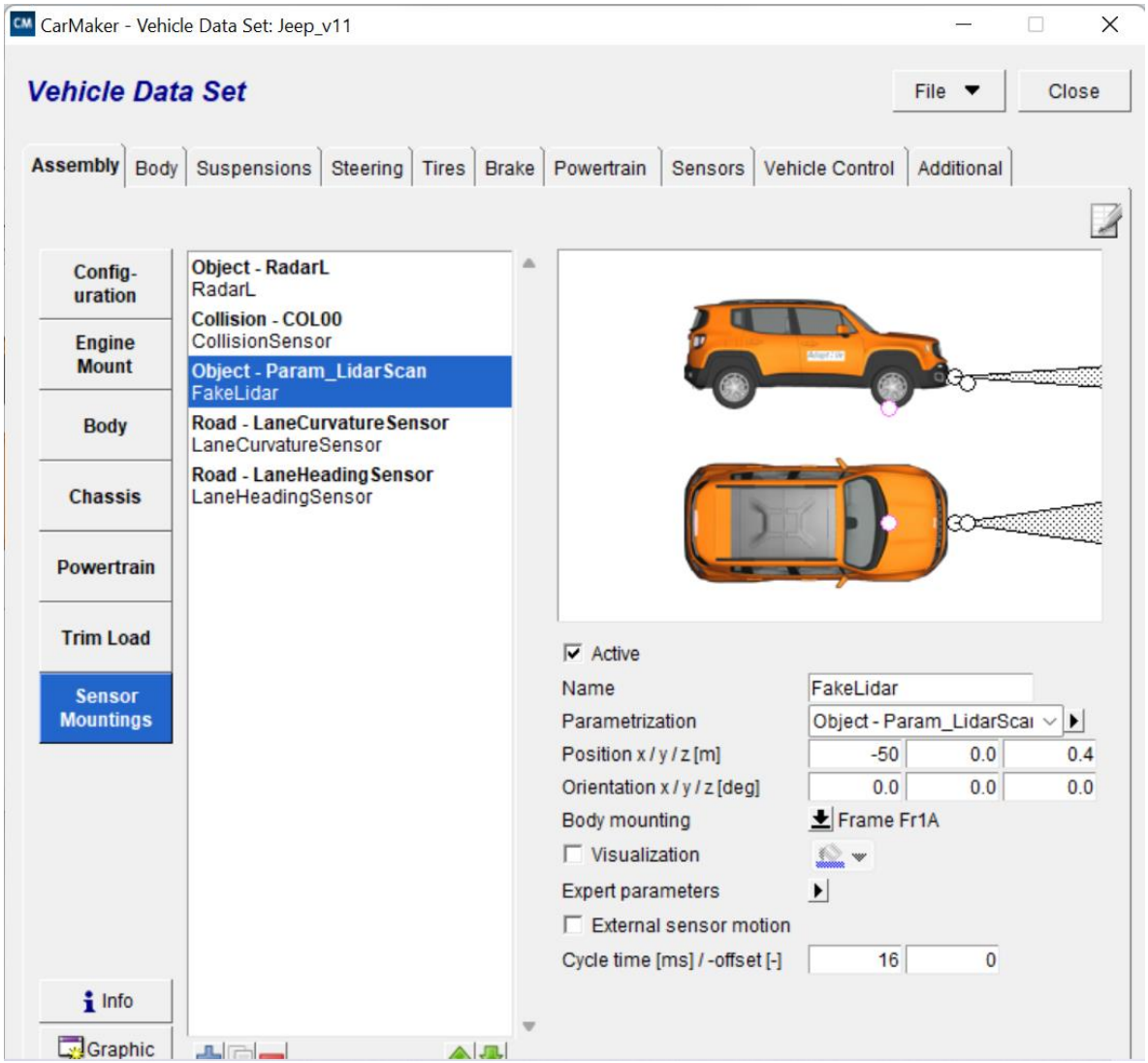

Figure 20: CarMaker vehicle editor.

Available components and modules

Basic components and modules that are associated with the **CarMaker** tool are:

- Office pro: related to the core simulation software for virtual test driving.
- Physical sensor models: related to physics-based sensor models like free space sensors, camera, radar, Lidar and Ultrasonic Raw Signal Interface.
- SimNet: is an extension to co-simulate multiple ego vehicles in one common scenario.

Self-driving agents

CarMaker can be used as a virtual environment for developing and testing the self-driving software. There are three different agents that can be used to control ego vehicles (either replacing the IPG driver or controlling additional ego vehicles for co-simulation with SimNet).

Within SUNRISE these agents will be taken as given self-driving systems to be tested in the use cases defined in WP7. The availability of three different self-driving agents allows to test the capacity of SUNRISE SAF to discover the weaknesses of given ADFs.

Following the above, the Dreams4Cars agent was developed in the H2020 Dreams4Cars project and further refined in follow up activities. It is a self-driving system that produces adaptive emergent behaviours with which it can solve situation never seen before and not programmed in advance [29].

A conventional (evolved) Motion Planner was also developed which is a planning algorithm that combines the strengths of sampling-based approaches with analytic optimal tree connections. This approach can compute feasible and safe obstacle-aware manoeuvres while exploring multiple local minima, setting it apart from conventional planning methods.

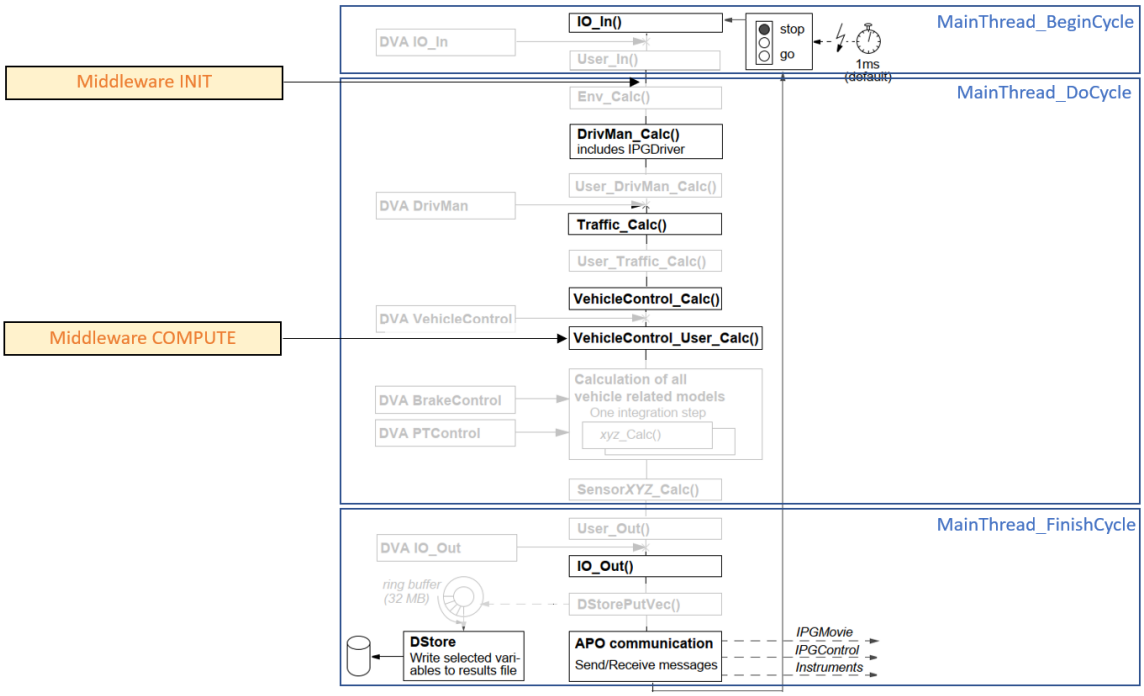

Figure 21: CarMaker Main cycle (src/CM_Main.c). Background picture from the CarMaker's programmers' guide.

Furthermore, a simple longitudinal controller is based on the Intelligent Driver Model (IDM). The implemented algorithm belongs to the well-known follow-the-leader deterministic models. The IDM offers a simple and efficient way to control the longitudinal dynamics of a vehicle. The principal advantages are the low computational cost and the robustness. However, the algorithm only shows reactive behaviours and relies on relative velocity and relative distance to the leading vehicle.

Middleware software

This is a software layer, compiled in **CarMaker** C++ that links the self-driving agents to the simulation engine. A block diagram is shown in Fig. 21. This software layer collects sensor data for the agent's input, calls the agent, and uses the agent's output to control the ego

vehicle. In addition, the middleware emulates V2V communications to co-simulate multiple ego vehicles in SimNet.

2.4 Tools for the subsystem “traffic agents”

In this section a review of the main existing simulation tools is provided which can support the “traffic agents” subsystem.

2.4.1 Simcenter Prescan

Actor Database

Simcenter Prescan comes default with large variety of actors (more than 75 models) including commercial cars, motors, trucks, buses, heavy machinery, emergency vehicles. Besides 3D models of the actors, actuators such as signal & braking lights, rotating steering & wheels also come as standard with **Prescan**.

Actor Motion

There are many ways to move or prescribe a trajectory to actuate an actor in **Prescan**:

- **Prescan** supports ASAM OpenDrive and OpenScenario. The user can import an ASAM OpenSCENARIO file into Prescan in which the motion of actors would be defined and used to move them.

- By using the **Prescan** GUI, the user can manually create a trajectory and assign an actor to that. Additionally, the speed profile can also be manually adjusted.

- **Prescan** also provides a Matlab / C++ API, through which the user can define trajectories of actors. This solution offers scalability when running test automation or creating trajectories for a large number of actors.

Traffic Simulators

Prescan comes with several plugins which allows users to co-simulate experiments with traffic simulators such as Aimsun, Vissim, and SUMO.

- Aimsun: This is a software tool that offers services for traffic planning, simulation and prediction. With the **Prescan** plugin, users can easily configure the experiment settings in their scenario and let Aimsun control all the configured agents that will not specifically take part of any pre-configured event in the simulation.

- Vissim: Similar to the Aimsun plugin, users can also configure the experiment settings in their scenario and let Vissim control the actors. It is also possible to inject **Prescan** controlled vehicles to Vissim simulations.

- SUMO: The SUMO plugin in **Prescan** is a tool that enables co-simulation and data transfer between the **Prescan** software and the Simulation of Urban MObility (SUMO)

software. This makes it possible to have traffic on a road network in a **Prescan** simulation that is generated by SUMO.

2.4.2 CarMaker

CarMaker includes several types of traffic objects (cars, truck, bikers, pedestrians, etc.) within a basic form of control (predefined, event-related trajectories).

In addition, via the SimNet plugin, several instances to **CarMaker** can be run in parallel and synchronised. In this way, more elaborate traffic agent controls are available: a vehicle in one **CarMaker** instance can be controlled with the IPG driver or one of the self-driving agents above. It then appears as an intentionally controlled and reactive vehicle in the other **CarMaker** instances.

2.4.3 CARLA

Actor database

CARLA includes an actor database [30] comprising various 3D models of vehicles, pedestrians, and props:

- Vehicles catalogue features a diverse range of models, including cars, trucks, vans, cyclists, motorcycles, and buses, as well as specialized vehicles, such as taxis, ambulances, and police cars.
- Pedestrian models encompass adults, children, and police officers.
- Props include static objects such as work signals, fences, and bags, which serve as obstacles and cannot be moved.

Vehicle agents

All vehicles are managed by the traffic manager by default, using autopilot to navigate randomly around the map. However, users can assign specific directions or waypoints for the vehicles to follow. **CARLA** also allows for the implementation of custom agents [31] for more controlled behaviors.

Several predefined agents are available, which operate based on environmental perception (using privileged information), waypoints, and a Proportional – Integral – Derivative (PID) controller. These agents follow a stop-and-go strategy when encountering obstacles. Custom implementations can replace any of these modules to suit specific needs or test independent modules.

Walker agents

Pedestrians in **CARLA** function differently from vehicles. There are two primary methods for pedestrian movement. The default method [32] involves specifying a starting point and an endpoint, with the shortest path used as the walking route. Alternatively, users can define custom agents [33] for pedestrians, similar to vehicle agents.

However, it is essential to distinguish walkable areas, such as sidewalks, from roads for effective pedestrian simulation. Users loading OpenDRIVE maps need to verify that the sidewalks tags are correctly loaded when using the default agent for moving the pedestrians [34].

Traffic simulators
CARLA supports multiple methods for defining traffic, independent of the agents used to move them:

- **CARLA** Scenario runner: This tool allows users to define traffic flows between two points, with vehicles driving on autopilot by default. It allows to define the scenarios either through its Python interface or through the ASAM OpenScenario standard.

- SUMO: **CARLA** supports co-simulation with SUMO to define complex traffic scenarios.

- Vissim: **CARLA** also supports co-simulation with Vissim, providing another option for detailed traffic simulation. A license is required to acquire the Driving Simulator Interface add-on.

2.5 Tools for the subsystem “connectivity”

In this section a review of the main existing simulation tools is provided which can support the “connectivity” subsystem.

2.5.1 CarMaker

To consider V2V communication, a software layer emulates Cooperative Awareness Messages (CAM messages). The communication link, beyond standard messages, includes the transmission of vehicle intentions (planned lateral and longitudinal manoeuvres) and the transmission of the curvilinear coordinate system in which the transmitted manoeuvres are to be interpreted. Both a primary and a secondary (desired) manoeuvre can be transmitted.

Armed with this information, every vehicle can predict with higher precision the intended (and desired) paths of the other collaborating one realizing efficient cooperative control (e.g. [35]). Multiple cooperative vehicles (each controlled by a different self-driving system) exchange V2V information in this way. Simulations of cooperative vehicles are carried out in **CarMaker** with the use of the SimNet pulg in. There is currently no model of the physical communication link (that is at the moment an ideal link).

As for what concerns I2V, the self-driving agents controlling the test vehicles can receive, in input, signals that represent future pahses and timings and, also, map information.

2.5.2 Simcenter Prescan

A V2X plugin in **Prescan** enables both V2V and V2I communications to be modelled and few common message types are supported such as ETSI CAM and DENM, and SAE BSM. Depending on the use case, required message types may be missing and would therefore need to be defined by a user and interfaced via C++/Simulink. For example, for a traffic light priority use case, commonly used message sets such as SREM and SPATEM are not included.

2.5.3 Network Simulator 3 (ns-3)

The network simulator 3 (**ns-3**) [36] is an open-source network simulation environment that aims at improving realism compared to the Network Simulator 2 (ns-2). **Ns-3** includes useful resources such as cross-layer features, scalability tools, and real-world integration features.

In the **ns-3** simulator each event is associated with its execution time governed by temporal increments. In this process, every event is associated to a point in simulation time where events are initiated and triggered consecutively, simulating discrete increments from one event to another.

Experimental research with **ns-3** simulator shows that realistic network models can be reproduced in wired and wireless systems. Moreover, the **ns-3** simulator provides models for network elements such as network nodes (end hosts, routers, switches, hubs, etc.), Ethernet and wireless links, and communication protocols.

Network simulator **ns-3** can be integrated with **CARLA** for mobility and sensor perception simulation and ns3 for network simulation. The network-**CARLA** simulator client module queries the information for the mobility of each of the **ns-3** simulated nodes and updates the Local Dynamic Map (LDM) module with all perception data sent over the simulated vehicular network.

2.6 Tools for the subsystem “simulation model validation”

In this section a review of the main existing simulation tools is provided which can support the “simulation model validation” subsystem.

2.6.1 CarMaker

CarMaker (and the related modules) is an industry-focused simulation environment. It includes models for vehicle dynamics and physical sensor models.

The models can be largely parametrized. The responsibility for correct parametrization is in the hands of the user, who is responsible for checking that the instantiated modules of a simulation describe the systems under test with acceptable approximations. This can be carried out with various tests that compare real world measurements to simulation outputs, for example to assess whether the two are statistically equivalent. An example of how to carry out such statistical comparisons in given in [37].

Some public materials with validations of various aspects of the **CarMaker** models have been published and related to the:

- validation of radar simulations with measuments [38].
- homologation of Electronic Stability Programme (ESP) systems [39].
- homologation of Electronic Stability Control (ESC) systems [40-42].
- homologation of ADFs [43].

2.6.2 CARLA

CARLA does not have a dedicated module for simulation verification. Instead, the responsibility of verifying the accuracy and correctness of the simulation lies with the user, who must utilize **CARLA**'s Python API. This section delineates the verifiable aspects and limitations within **CARLA**'s simulation environment to validate the simulations.

Camera sensors
Camera sensors within **CARLA** exhibit varying quality levels, as detailed in subsection 2.2.2. Ensuring that these sensors emulate real-world camera behavior accurately is a complex task.

Through **CARLA**'s Python API, users can examine the content captured by the camera, including object segmentation and image properties such as resolution, pixel dimensions, image format, etc. Nonetheless, the simulation cannot replicate real physical camera phenomena such as noise, distortions, and temperature-induced effects.

Distance-based sensors
Distance-based sensors, including Radar and LIidar, employ the z-buffer from the renderer to measure the exact distance to objects. These sensors' parameter configurations mirror those of their physical counterparts, allowing users to validate sensor data by extracting information from the **CARLA** environment.

However, these sensors do not interact with the material properties of target objects, as **CARLA** only alters the visual appearance of objects. Consequently, material characteristics like reflectance are not considered and cannot be simulated via the Python API.

Vehicle dynamics
To assess vehicle dynamics, users can issue commands to actuators either directly or through the default agents described in subsection 2.3.2. **CARLA** facilitates the extraction of all actuator outputs and vehicle movements, enabling comparisons with real-world vehicle physics.

For high-fidelity simulations, it is advisable to configure the chrono vehicle as outlined in subsection 2.3.2, which offers a more precise representation than other setups. Additionally, surface friction parameters should be adjusted to reflect real testing conditions accurately.

Scenarios and behaviour agents
One of the most critical aspects of a simulation is the environment, including the simulated scene and the behaviors of the simulated agents (e.g., pedestrians, vehicles, vulnerable road users). The simulation environment should closely match the scenario description file. However, during the execution of **CARLA**'s scenario runner, only driving statistics are provided. To verify the simulated scenario, users can extract all agent information and perform verification using post-processing scripts.

2.6.3 WayWiseR

Building upon the foundation laid by WayWise, **WayWiseR** integrates WayWise with ROS2 (Robot Operating System 2). **WayWiseR** enhances the utility of WayWise by providing an abstraction layer between WayWise and ROS2, along with necessary launch files and configuration settings.

This integration facilitates the utilization of ROS2-based open-source stacks for navigation and perception, thereby augmenting the validation process. Moreover, the abstraction layer implemented by **WayWiseR** promotes a high degree of code reusability between real vehicle implementations and simulations. This approach significantly enhances the validation process by allowing for the comparison of simulation results with real-world experimental data.

3 SIMULATION TOOLING SPECIFICATIONS ASSIGNED TO SUBSYSTEM REQUIREMENTS

Overall, ensuring high model fidelity of the subsystems of the simulation framework presented in the deliverable D4.1 is essential for generating reliable and actionable results, reducing the need for physical testing, and accelerating the development and optimization of automotive systems. This chapter aims to provide simulation tooling specifications with regards to the subsystem requirements with respect to tools, interfaces, and model fidelity, which were presented in the chapter 4 of the deliverable D4.1.

Firstly, most of the tools presented in chapter 2, support test automation tools requirements, such as automated execution of simulation tests, test case generation, and result analysis, as well as modelling and simulation tools requirements, such as the capability of creating realistic virtual models of automotive systems, vehicle dynamics, powertrain, control systems, and environmental conditions. For example, Simcenter Prescan is the main simulation tool providing high fidelity environment & sensor simulation.

Additionally, these tools can also support the requirements management tools (capture, tracking, and traceability), the data management and analysis requirements (data storage, retrieval, and post-processing), and the version control, configuration management and reporting tools requirements (help, manage, and track changes to simulation models, test cases, and related artifacts, ensuring the requirements related to proper versioning, collaboration, and traceability).

Coming to the interface requirements, most of the tools presented in chapter 2 use open standards for the interfaces (Open SCENARIO, OpenDRIVE, etc.) between the SAF and the simulation framework, as well as the interfaces between the subsystems. In particular, the environment simulation tool IPG CarMaker was analysed in terms of its ability to interface with the targeted high-fidelity radar models.

Finally, most of the tools presented in chapter 2 support the model fidelity requirements (geometric fidelity, material properties, sensor and actuator models, functional fidelity), where the virtual models used in simulations and analyses reproduce the state and behaviour of a real-world object, feature, or condition. For example, in terms of sensor fidelity, CARLA has compatibility for 3d-party software to define the sensors used.

In the light of the above, it can be stated that the tools included in chapter 2 cover the subsystem requirements from a SAF perspective (D4.1 chapter 4) with respect to tools, interface, and model fidelity requirements.

4 SIMULATION TOOLING SPECIFICATIONS ASSIGNED TO UC REQUIREMENTS

The simulation of CCAM functionalities in UCs is of crucial importance and requires dedicated models and tools for the vehicles, sensors and environment, with the aim to be highly representative of the real world. Simulation tooling specification has a key role to play in virtually investigating the behavior of CCAM systems under test [44].

In this context, the present study aims to provides specific simulation toolings for the safety validation testing of the CCAM systems presented in the defined use cases in the deliverable D7.1.

4.1 Urban AD perception validation (UC 1)

The scope of UC 1 - Urban AD perception validation is to validate the environment perception for SAE L3+ vehicles in urban and/or suburban areas, using a hybrid validation approach (virtual simulations and physical tests). Also, aspects of connected driving and collective perception are considered in this use case.

UC 1 - Urban AD perception validation includes three main sub-UCs as follows:

- sub-UC 1.1 - perception testing: covers sensor models used in the perception AD subsystem of an urban chauffer.

- sub-UC 1.2 - connected perception testing: builds on sub-UC 1.1 and covers the integration of information from other vehicles/VRUs coming from external sources via V2X and the use of C-ITS services, such as Cooperative Awareness Messages (CAM), Distributed Environmental Notification Messages (DENM), etc.

- sub-UC 1.3 - cooperative perception testing: builds on sub-UC 1.2 and covers the integration of information from other vehicles/VRUs coming from external sources via V2X and the use of C-ITS services, such as Collective Perception Systems (CPMs).

In the following subsections 3.1.1, 3.1.2 and 3.1.3, simulation tooling specifications will be provided, based on the defined subsystem requirements in D4.2, in order to validate the CCAM systems presented in the above sub-UCs.

4.1.1 Perception testing (sub-UC 1.1)

4.1.1.1 Short description

Sub-UC 1.1 aims to cover the testing of the different elements of the perception layer when the ADS operational design domain includes complex urban intersections and the inclusion of adverse weather conditions. The scope is to extend the current possibilities of testing CCAM

functions in urban environments focusing on intersections where the majority of accidents occurs between cars and pedestrians.

In this context, a representative perception AD subsystem is addressed which is based on three different sensors (Lidar, camera and radar). Based on the work in the deliverable D4.2 (chapter 5), forty-six (46) requirements were identified in total for sub-UC 1.1 and mapped to the subsystems of the SUNRISE Harmonized V&V Simulation Framework. These requirements are a refined version of the ones defined in the deliverable D7.1 and are presented below:

- Six (6) requirement IDs (R1.1_01, R1.1_02, R1.1_16, R1.1_26, R1.1_27, R1.1_28) are related to the “test case manager” subsystem.

- Six (6) requirement IDs (R1.1_06_X(group), R1.1_07_X(group), R1.1_12_X(group), R1.1_22, R1.1_23, R1.1_24) are related to the “environment” subsystem.

- Twenty-two (22) requirement IDs (R1.1_03, R1.1_03_01, R1.1_03_02, R1.1_03_03, R1.1_03_04, R1.1_03_05, R1.1_03_06, R1.1_03_07, R1.1_03_08, R1.1_03_09, R1.1_04_01, R1.1_04_02, R1.1_04_03, R1.1_04_04, R1.1_04_05, R1.1_04_06, R1.1_04_07, R1.1_05, R1.1_11, R1.1_11_01, R1.1_11_02, R1.1_26) are related to the “subject vehicle’s sensors” subsystem.

- Four (4) requirement IDs (R1.1_09, R1.1_10, R1.1_13, R1.1_20) are related to the “subject vehicle’s AD functions” subsystem.

- One (1) requirement ID (R1.1_21) is related to the “subject vehicle’s dynamics” subsystem.

- Four (4) requirement IDs (R1.1_08_01, R1.1_08_02, R1.1_18, R1.1_19) are related to the “traffic agents” subsystem.

- Three (3) requirement IDs (R1.1_14, R1.1_24, R1.1_25) are related to the “simulation model validation” subsystem.

All the individual partners (VIF, IFAG, SISW, CVC) participating in the present sub-UC contributed to its respective tooling specification with the aim all relevant requirements to be met by the proposed S/W tools. It should be stated that the availability of licenses and knowledge of the tools were part of the decisions.

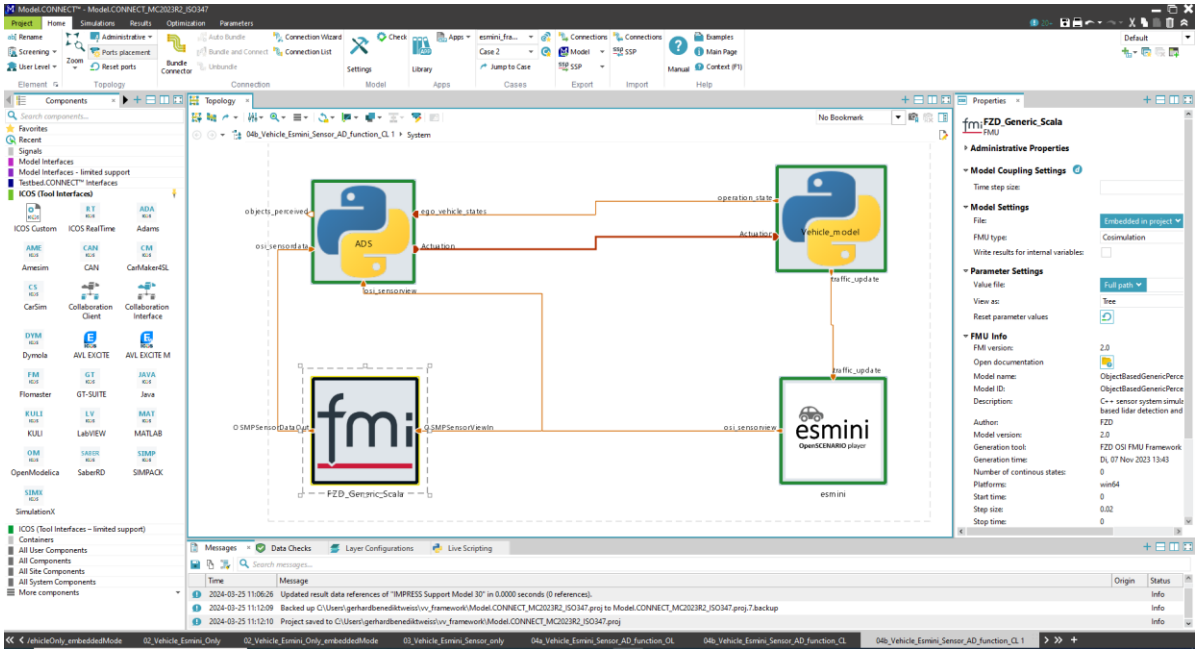

Figure 22: Co-Simualtion Topology of sub-UC1.1 in Model.CONNECT

Figure 22 depicts the simulation framework implemented by ViF. Following the information provided in Chapter 2, Model.CONNECT™ tool is used as «test case manager» since it is a neutral co-simulation platform with the possibilties to run (test) cases and postprocess functions. Moreover, it serves also as master for the sensor FMU. Furthermore, Esmini tool is used as minimalistic environment simulator as there is no need for a photorealstic environment simulation in this tooling. Moreover, Esmini includes SUMO (Simulation of Urban MObility) which is used for the traffic agent simulation.

An open source highly parameterizable generic perception sensor and tracking model [45] is used as FMU. It has an ASAM OSI Interface and is parametrized as LiDAR model. A python module, named RoMPaC (Robust Motion Planning and Control), is used for simulating ADs together with a vehicle model. For the sake of modularity RoMPaC has been spilt into AD-function simulation and vehicle dynamics simulation.

Figure 23 depicts the simulation framework implemented by IFAG, which comprises two principal subsystems. The first is the radar sensor model, realised as an FMU, which is interconnected with the environment simulation, represented by the IPG CarMaker simulation software. In addition, IPG CarMaker acts as a test case manager, providing the targeted test cases related to EuroNCAP benchmark scenarios and evaluation functionalities to analyse the set metrics and KPIs. Moreover, it serves as an FMU-Master for the radar sensor FMU. In addition to the environment, IPG CarMaker provides vehicle dynamics and traffic agents simulation capabilities to the simulation framework. Finally, the high-fidelity radar sensor model, including detailed parameterisation capabilities, is realised via an FMU following the ASAM OSI Interface specification.

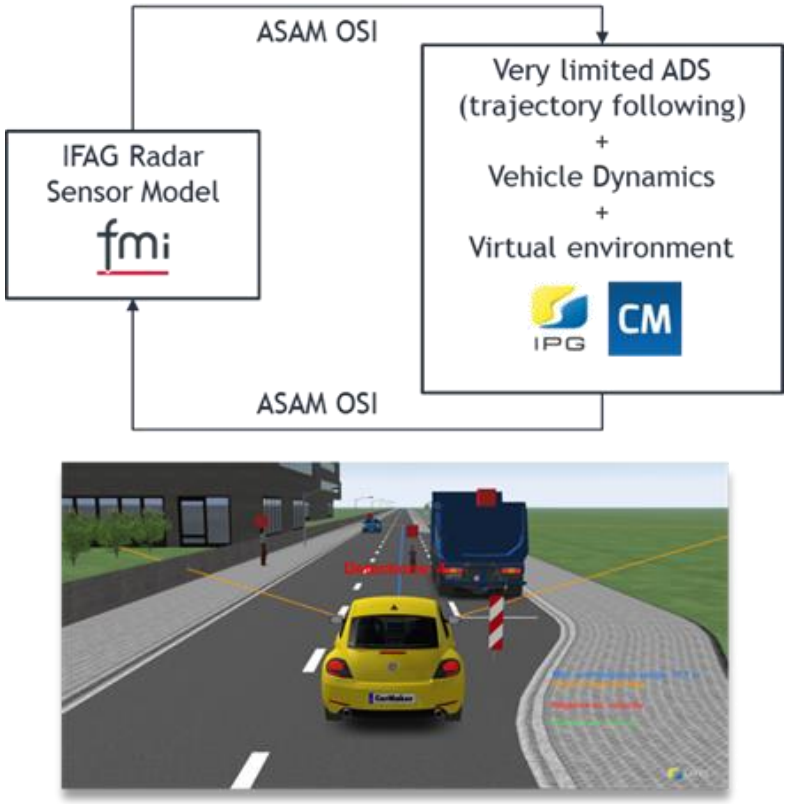

Figure 23: Simulation framework for the radar sensor model with respect to sub-UC 1.1.

With respect to the simulation framework implemented by SISW, as presented in Chapter 2, Simcenter Prescan is the main simulation tool providing high fidelity environment & sensor simulation. The simulation environment provides interfaces for vehicle dynamics co-simulation, traffic agent simulation, and AD function testing. Additionally, Simcenter HEEDS and Prescan scene editor tools provide test orchestration, test management, and a methodology for unknown-unsafe scenario exploration.

Finally, wth respect to the simulation framework implemented by CVC, it should be stated that the system under evaluation relies on cameras, and thus, realistic camera images are necessary. To generate these images and assess the model's performance, a realistic rendering tool will be employed. Errors identified in this process will be propagated to CARLA to evaluate the driving maneuvers in the scenarios. Moreover, Scenario Runner manages the 'Test Case Manager' subsystem, creating the driving agent in CARLA clients to interact with the CARLA server and its environment. Additionally, it provides all necessary information to validate the correctness of the simulation. The environment is loaded from CARLA, either using the default assets created in CARLA or OpenDrive. CARLA also supports OpenScenario for specifying the characteristics of scenarios. However, it should be noted that in this simulation set up, the presence of weather effects, such as rain, is not used to adapt the friction with the road accordingly. Traffic agents and their behaviours are loaded from CARLA assets according to subchapter 2.4.3. The behaviours of these traffic agents will follow the CARLA autopilot, while the ego-vehicle will be controlled by a defined agent developed by the CVC using machine learning models.

Mapping of the used simulation tools for each subsystem of the simulation framework is given in Table 2.**Error! Reference source not found.**

Table 2: Mapping of the simulation framework subsystems to specific simulation tools in sub-UC 1.1.

Subsystem	Simulation tools
Test case manager	Model.CONNECT™ IPG CarMaker Simcenter HEEDS Simcenter Prescan scene editor CARLA Scenario runner
Environment	Esmini IPG CarMaker Simcenter Prescan CARLA + Phtoto-realistic renderer
Subject vehicle - Sensor	OBGPOM FMU 2.0 (genereted from C code) IFAG Radar Sensor FMU 2.0 Simcenter Prescan CARLA (Unreal)
Subject vehicle – AD function	Python/RoMPaC IPG CarMaker Interface via C++ / Simulink Python / CARLA
Subject vehicle – Vehicle dynamics	Python/RoMPaC IPG CarMaker Simcenter Prescan / FMI-FMU CARLA (Unreal)
Traffic agents	esmini using SUMO IPG CarMaker Simcenter Prescan CARLA
Connectivity	Not applicable for the present sub-UC
Simulation model validation	Python + CARLA – Scenario runner

4.1.1.2 Mapping of subsystem requirements and corresponding S/W features

Table 3 lists the requirements for the simulation tools, as these were provided in D4.2, and how they are met by the selected simulation tools.

Table 3: Subsystems, requirements and S/W tool features for the simulation tooling specification of the sub-UC 1.1.

Subsystem	Req number ID	S/W tool name	S/W tool feature that meets the requirement
Test Case Manager	R1.1_01	Model.CONNECT / Simcenter Prescan	Model.CONNECT supports a postprocess script can be (automatically) executed to calculate (any) KPIs

			Simcenter Prescan supports a library of EuroNCAP protocol scenarios including generation of KPIs
	R1.1_16	CARLA Scenario Runner	Co-simulation platform, case execution, OpenScenario support
	R1.1_26	Simcenter Prescan / CARLA	Simcenter Prescan supports Matlab/Python/C++ API or Prescan GUI CARLA support this requirement through the python API
	R1.1_27	Simcenter Prescan	Supports local saving of video data
	R1.1_28	Model.CONNECT / Simcenter HEEDS	Model.CONNECT uses a postprocess script that can be (automatically) executed to calculate (any) KPIs Simcenter HEEDS can be used as an analysis and visualization tool. User-defined postprocessing may also be added.
Environment	R1.1_06_X(grroup)	Model.CONNECT / Esmini / Simcenter Prescan / CARLA	Model.CONNECT provides paths to OpenScenario files that can be definded in the test cases and passed to subsystems. Esmini supports OpenDrive format. Many ODD attributes are supported by Simcenter Prescan and CARLA.
	R1.1_07_X(group)	Esmini / Simcenter Prescan / CARLA	Esmini supports OpenScenario format Wind, fog, and illumination are supported by Simcenter Prescan Wind is not supported by CARLA
	R1.1_12_X(group)	Esmini	supports OpenScenario format
	R1.1_22	Simcenter Prescan / CARLA	Simcenter Prescan supports import of Opendrive, large asset and environment models CARLA supports OpenScenario and OpenDrive formats. It has some default maps with assets that represent urban environments.
	R1.1_23	Simcenter Prescan / CARLA	Simcenter Prescan supports physics based camera (Unreal Engine)

			CARLA supports high-fidelity camera images can be generated, although not at real time
Subject vehicle - Sensor	R1.1_05	OBGPOM FMU 2.0	the sensor model is highly parametzriable
	R1.1_26	OBGPOM FMU 2.0	several sensor profiles including mounting position can be created in advance and selected via the test case manager
	R1.1_03	FMU / Simcenter Prescan	High-fidelity radar sensor model (FMU) Detailed Radar Sensor & Physics Based Radar Sensor (Simcenter Prescan)
	R1.1_03_01	FMU / Simcenter Prescan	High-fidelity radar sensor model (FMU)
	R1.1_03_02	FMU / Simcenter Prescan	High-fidelity radar sensor model (FMU) Physics Based Radar Sensor (Simcenter Prescan)
	R1.1_03_03	FMU / Simcenter Prescan	High-fidelity radar sensor model (FMU)
	R1.1_03_04	FMU / Simcenter Prescan	Detailed Radar Sensor & Physics Based Radar Sensor (Simcenter Prescan)High-fidelity radar sensor model (FMU)
	R1.1_03_05	IPG CarMaker / FMU / Simcenter Prescan	Environment simulation (CarMaker) High-fidelity radar sensor model (FMU) Detailed Radar Sensor & Physics Based Radar Sensor (Simcenter Prescan)
	R1.1_03_06	FMU	High-fidelity radar sensor model (FMU)
	R1.1_03_07	FMU / Simcenter Prescan	High-fidelity radar sensor model (FMU) Detailed Radar Sensor (Simcenter Prescan)
	R1.1_03_08	FMU / Simcenter Prescan	High-fidelity radar sensor model (FMU)

			Detailed Radar Sensor & Physics Based Radar Sensor (Simcenter Prescan)
	R1.1_03_09	IPG CarMaker / FMU	Environment simulation (CarMaker) High-fidelity radar sensor model (FMU)
	R1.1_04_01	Simcenter Prescan / CARLA (Photorealistic renderer)	Simcenter Prescan supports Camera Sensor & Physics Based Camera (Unreal) CARLA supports RGB Camera Sensor
	R1.1_04_02	Simcenter Prescan / CARLA (Photorealistic renderer)	Camera Sensor & Physics Based Camera (Unreal) CARLA supports RGB Camera Sensor
	R1.1_04_03	Simcenter Prescan / CARLA (Photorealistic renderer)	Camera Sensor & Physics Based Camera (Unreal) CARLA supports RGB Camera Sensor
	R1.1_04_04	Simcenter Prescan / CARLA (Photorealistic renderer)	Camera Sensor & Physics Based Camera (Unreal) CARLA supports RGB Camera Sensor
	R1.1_04_05	Simcenter Prescan / CARLA (Photorealistic renderer)	Camera Sensor & Physics Based Camera (Unreal) CARLA supports RGB Camera Sensor
	R1.1_04_06	Simcenter Prescan / CARLA (Photorealistic renderer)	Simcenter Prescan supports sensor pose (can be set by user freely) CARLA supports sensor pose
	R1.1_04_07	Simcenter Prescan	Default format RGB -> User has to convert as post-process
	R1.1_05_X(group)	Simcenter Prescan	Point Cloud Lidar Sensor
	R1.1_11_01	Simcenter Prescan / CARLA	Object camera sensor / Bounding box sensor (Simcenter Prescan) Obstacle detector sensor, segmentation sensors or using the python API (CARLA)
	R1.1_11_02	Simcenter Prescan / CARLA	Legacy lane marker sensor (Simcenter Prescan)

			Python API (CARLA)
	R1.1_26	CARLA / Photorealistic renderer	Sensor pose + sensor parameters
Subject vehicle – AD function	R1.1_09	Python/RoMPaC / Simcenter Prescan / CARLA Scenario Runner	the class urban pilot and the class robotaxi support several driving manoeuvres (Python/RoMPaC) Scenarios can be defined by user through OpenScenario, Prescan APIs, or manually on GUI. Scenarios in CARLA scenario runner can be defined by the user using the default maps or by Openscenario
	R1.1_10_X(group)	Simcenter Prescan	Scenarios mentioned in the requirements can be performed.
	R1.1_20	Python/RoMPaC / CARLA	the AD function works with objects lists independent of the sensor type (Python/RoMPaC) Autopilot (CARLA)
Subject vehicle – Vehicle Dynamics	R1.1_17	Simcenter Prescan / CARLA	User can configure sensor position freely via GUI or API (Simcenter Prescan) Python API allows to place sensors at any place (CARLA)
	R1.1_21	Python/RoMPaRC / Simcenter Prescan / CARLA	Python/RoMPaRC includes a vehicle model that can be parametrized User can set speed of SuT freely via API (Simcenter Prescan) Target speed (CARLA)
Traffic Agents	R1.1_08_01	Esmini - SUMO / Simcenter Prescan / CARLA	supports OpenScenario format (Esmini – SUMO) All actors defined in the requirement are supported in Prescan All actors are supported (CARLA)
	R1.1_08_02	esmini/SUMO	no special agentes are foreseen

	R1.1_18	Esmini - SUMO / Simcenter Prescan / CARLA	a controller for the traffic agents can be definded by the OpenScenario format which is supported by Esmini Behaviors must be defined by user through OpenScenario, Prescan APIs, or manually on GUI. Manoeuvres has to be specified by the user (CARLA)
	R1.1_19	Esmini - SUMO / Simcenter Prescan / CARLA	a controller for the traffic agents can be definded by the OpenScenario format which is supported by Esmini Behaviors must be defined by user through OpenScenario, Prescan APIs, or manually on GUI. Manoeuvres has to be specified by the user (CARLA)
Simulation model validation	R1.1_25	CARLA Scenario Runner	All agents behaviours are saved. A postscript can be used to compare the desired and performed manoeuvres.

4.1.1.3 Limitations

It can be concluded from Table 3 that Simcenter Prescan and HEEDS are suitable tools for sub-UC1.1 and may be enhanced or used together with other tools to achieve a complete simulation framework and meet all the defined requirements in the deliverable D4.2 (section 8). Most of the requirements for sub-UC1.1 for the following subsystems

- Test case manager
- Environment
- Subject vehicle
- Traffic Agents

are met by the two tools. A few (sub) requirements are not met by the two tools for these subsystems – the explanations for these are as follows:

- R1.1_11: The requirement relates to the SuT perception algorithm with respect to its capabilities, Therefore, it is not related to simulation tooling.
- R1.1_02: Radar validation metrics – metrics not available, may be defined by user and added to Simulink / C++ processing or post-processing.

- R1.1_24: Simulation model validation – Photo-realism checks – Photorealism check is not provided, would need a feature addition.

- R1.1_03_03, R1.1_03_06, and R1.1_03_09 – all relate to compatibility of radar sensor model with ASAM Open Simulation Interface (OSI) format. OSI is not yet supported by Prescan. The OSI support would need to be added to the software to meet these requirements.

In addition, Simcenter Prescan and HEEDs cannot fulfill the requirements for sub-UC1.1 related to the subsystem “Simulation model validation”. For these requirements, the tool would need to be enhanced with features or used together with other tools which (i) verify that the performed simulation is correct with respect to the test scenario, and (ii) support correlation studies between different test benches in order to assess fidelity of the simulation models.

Additionally, CARLA platform does not meet the requirements for high-quality realism necessary for evaluating the perception models of the cameras. In such cases, an object list will be used, injecting noise based on evaluations conducted on photo-realistic static data generated for these scenarios. This photo-realistic data can be parameterized according to the real camera setup and urban environments.

The following points define the requirements accomplished by each tool:

CARLA Scenario Runner:

- Scenario handling: R1.1_16

- Agents manoeuvers: R1.1_09

- Scenario evaluation: R1.1_25. The tool only provides all the information to evaluate the driving. It has to be done using a post-process script.

Photo-realistic render:

- Camera sensor: R1.1_23, R1.1_04_01, R1.1_04_02, R1.1_04_03, R1.1_04_04, R1.1_04_05, R1.1_04_06, R1.1_26. Camera parameters and position can be modified during the generation.

CARLA:
- Environment: R1.1_22, R1.1_06, R1.1_08_1. All ODD cases are supported except for wind. Also, the water particles and wet patches are only visual, they do not have any physical effect on the driving.

- Subject vehicle / sensors: R1.1_04_01, R1.1_04_02, R1.1_04_03, R1.1_04_04, R1.1_04_05, R1.1_04_06, R1.1_26. All camera poses and parameters are supported.

Although the camera is a simplified version of a real camera in CARLA. R1.1_23 is not accomplished in CARLA.

- Subject vehicle / objects: R1.1_11_01, R1.1_11_02

- Subject vehicle / behaviour: R1.1_20, R1.1_17, R1.1_21, R1.1_18, R1.1_19

4.1.2 Connected perception testing (sub-UC 1.2)

4.1.2.1 Short description

The sub-UC 1.2 considers the use of the SUNRISE SAF for testing cooperative perception and decision making in urban intersection scenarios. The SuT will be a Cooperative-ACC that is expected to improve when enhanced perception is obtained by V2X connectivity.

More in detail, four functional scenarios are defined for the sub-UC 1.2 in the deliverable D7.1. The first three scenarios deal with improving vehicle behaviour when the vehicle approaches a traffic light and receives traffic light phases, timings and map information, and the fourth scenario deals with a red light violation of a crossing car.

Within the SUNRISE project, three partners (VED, IKA and UNITN) are involved in sub-UC 1.2. VED and IKA will mainly support the physical tests, whereas UNITN will run simulations in IPG CarMaker simulation environment with a model of the Cooperative-ACC SuT.

Based on the work in the deliverable D4.2 (chapter 8, table 4), thirty-one (31) specific requirements were identified in total for sub-UC 1.2, based on the relevant subsystems of the SUNRISE Harmonized V&V Simulation Framework. These requirements are a refined version of the ones defined in the deliverable D7.1 and are presented below:

- One (1) requirement ID (R1.2_01) is related to the “test case manager” subsystem.
- Nineteen (19) requirement IDs (R1.2_02_01, R1.2_02_02, R1.2_02_03, R1.2_02_04, R1.2_02_05, R1.2_02_06, R1.2_02_07, R1.2_02_08, R1.2_02_09, R1.2_02_10, R1.2_02_11, R1.2_02_12, R1.2_02_14, R1.2_02_15, R1.2_02_16, R1.2_02_17, R1.2_03_01, R1.2_03_02, R1.2_03_04) are related to the “environment” subsystem.
- Ten (10) requirement IDs (R1.2_05, R1.2_06_01, R1.2_06_02, R1.2_06_03, R1.2_06_04, R1.2_07, R1.2_08_01, R1.2_08_02, R1.2_08_03, R1.2_08_04) are related to the “subject vehicle” subsystem.
- One (1) requirement ID (R1.2_04_01) is related to the “traffic agentst” subsystem.

4.1.2.2 Mapping of subsystem requirements and corresponding S/W features

Table 4 lists the requirements for the simulation tools, as these were provided in D4.2, and how they are met by the simulation systems.

Table 4: Subsystems, requirements and S/W tool features for the simulation tooling specification of the sub-UC 1.2.

Subsystem	Req number ID	S/W tool name	S/W tool feature that meets the requirement
Test case manager	R1.2_01	CarMaker	In principle any simulation parameter and state is measurable (but the KPI should be specified).
Environment	R1.2_02_X (X = 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 14, 15, 16, 17)	CarMaker with SIMNET and V2V emulation	Simulations can be set up meeting all requirements with respct to scenery description. Minor limitations for the requirement R1.2_02_10.
	R1.2_03_01	Carmaker	Wind can be included in CarMaker simulations, although not seems to be relevant for the test scenarios in the sub-UC 1.2.
	R1.2_03_02	CarMaker	Different illuminations are possible in CarMaker simulations, although not seems to be be relevant for the test scenarios in the sub-UC 1.2. Current set up uses high-level sensor data objects list and abstracts from visibility issues.
	R1.2_03_04	CarMaker	V2V is emulated with a custom software layer on top of SimNet. Only V2V is available. I2V messages can be pre-programmed. There is no low-level physical model of the communication link.
Subject Vehicle	R1.2_05	CarMaker	CarMaker supports three different self-driving agents; co-driver, motion planner and simple controller based in the IDM. The former two supports all types of longitudinal and lateral maneuvers (IDM only longitudinal ones) [46].
	R1.2_06_X (X = 01, 02, 03, 04)	CarMaker	All three agents above (with limitations for the IDM) produce emergent behaviour when approaching a traffic light with known phases. The co-driver and motion planner reacts to obstacles with declared trajectory. The DENM message must be programmed before the simulation is run. However, if the message is late, there may be no escape maneuver.
	R1.2_07	CarMaker	Car Maker can meet all the perception DF requirements except ODD boundary transitions (if a vehicle can't drive in a given situation it might also be uncapAble of detecting the reason why the situation

			is outside its ODD). SpaT Map and DENM messages must be pre-programmed currently.
	R1.2_08_X (X = 01, 02, 03, 04)	CarMaker	The way in which the detection layer is downgraded by adverse conditions is not modelled at low-level physics. The effect of adverse conditions (e.g., losses and delays) must be defined and programmed in the V2V emulation layer.
Traffic agents	R1.2_04_01	CarMaker	CarMaker supports all listed types of objects, as those described in this requirement.

4.1.2.3 Limitations

It can be concluded from Table 4 that CarMaker is a suitable tool for sub-UC1.2 and may be enhanced or used together with other tools to achieve a complete simulation framework and meet all the defined requirements in the deliverable D4.2 (section 8). There are some minor limitations that can be handle with workarounds, by means of overcoming simulation limitations or/and constraints, such as:

a) KPIs should be specified for the present subUC 1.2 to be sure that can be met by the CarMaker tool;

b) CarMaker may not be able to meet all the types of the drivable area signs (regulatory, warning, or information), as described in the requireMent R1.2_02_10;

c) communications are modelled at the functional level and there is no low-level physical model of the communication link;

d) CarMaker can not meet the perception DF requirement about ODD boundary transitions;

e) the SuT in the simulations is not exactly the same that will be tested on proving grounds.

4.1.3 Cooperative perception testing (sub-UC 1.3)

4.1.3.1 Short description

The following analysis relates to the specification of a simulation tooling for the sub-UC 1.3, which considers the use of the SUNRISE SAF for the validation of the urban cooperative environment perception layer system, within a predefined ODD context. This system shall be capable of communicating with sensor-equipped Road-Side Units (RSUs) and Vulnerable Road Users (VRUs), by aggregating cooperative awareness information in Collective Perception Messages (CPMs).

In order to evaluate the benefits provided by the exchange of such messages, a simulation tooling framework was built up to jointly, accurately model the mobility, generated sensor data, and communication over the wireless channel. This framework integrates CARLA and ns-3 simulation tools. CARLA is mainly used for the simulation of vehicle mobility, but also sensor data from CARLA are encoded and send CPMs over a simulated network on ns-3.

Based on the work in the deliverable D4.2 (chapter 5), ten (10) specific requirements were identified in total for sub-UC 1.3, based on the relevant subsystems of the SUNRISE Harmonized V&V Simulation Framework. These requirements are a refined version of the ones defined in the deliverable D7.1 and are presented below:

- Two (2) requirement IDs (R1.3_01, R1.3_10) are related to the “test case manager” subsystem.

- Four (4) requirement IDs (R1.3_02, R1.3_07, R1.3_08, R1.3_09) are related to the “environment” subsystem.

- One (1) requirement ID (R1.3_06) is related to the “subject vehicle” subsystem.

- Three (3) requirement IDs (R1.3_03, R1.3_04, R1.3_05) are related to the “connectivity” subsystem.

Moreover, after refining the test objectives for sub-UC 1.3 during the activities of the present task, additional requirements were identified, on top of the ones defined in the deliverables D7.1 and D4.2. These additional requirements are denoted as “N-”, their descriptions can be found in annex 2, and are presented below:

- Three (3) requirement IDs (N-R1.3_11, (N-R1.3_12, N-R1.3_13) are related to the “test case manager” subsystem.

- One (1) requirement ID (N-R1.3_14) is related to the “environment” subsystem.

- Fourteen (14) requirement IDs (N-R1.3_15, N-R1.3_16, N-R1.3_17, N-R1.3_18, N-R1.3_19, N-R1.3_20, N-R1.3_21, N-R1.3_22, N-R1.3_23, N-R1.3_24, N-R1.3_25, N-R1.3_26, N-R1.3_27, N-R1.3_28 are related to the “subject vehicle” subsystem.

- Two (2) requirement IDs (N-R1.3_29, N-R1.3_30) are related to the “traffic agents” subsystem.

- Two 2) requirement IDs (N-R1.3_31, N-R1.3_32) are related to the “connectivity” subsystem.

- One (1) requirement ID (N-R1.3_33) is related to the “simulation model validation” subsystem.

4.1.3.2 Mapping of subsystem requirements and corresponding S/W features

All the above requirement IDs for the simulation tooling specification of the sub-UC 1.3 are depicted in Table 5 in relation to the subsystems of the simulation framework, the S/W tool names and the S/W tool features, which meet these requirements.

Table 5: Subsystems, requirements and S/W tool features for the simulation tooling specification of the sub-UC 1.3.

Subsystem	Req number ID	S/W tool name	S/W tool feature that meets the requirement
Test case manager	R1.3_01	CARLA simulator	CARLA provides ground truth data which can be used together with real measurements.
	R1.3_10	CARLA simulator	CARLA supports OpenScenraio format which reads a test scenario description file and generates a corresponding scenario environment.
	N-R1.3_11	CARLA simulator	CARLA has a configuration interface allowing the user to configure at least the sensor position and its basic parameters.
	N-R1.3_12	CARLA simulator	CARLA can store at least 1 hour of video data from the camera.
	N-R1.3_13	CARLA simulator	CARLA has a dedicated module in analysing the scenario results and generating metrics for the test coverage evaluation.
Environment	R1.3_02	CARLA simulator	CARLA includes Python API in supporting SiL and CoSim virtual tests.
	R1.3_07	CARLA simulator	CARLA supports the ODD/Scenery elements for all the examined test scenarios.
	N-R1.3_14	CARLA simulator	CARLA can visually represent a cooperative urban environment.
	R1.3_08	CARLA simulator	CARLA supports the ODD/Atmospheric conditions elements for all the examined test scenarios.
	R1.3_09	CARLA simulator	CARLA supports the ODD/Dynamic elements (traffic participants) behaviour elements for all the examined test scenarios.
Subject vehicle / sensors	N-R1.3_15	CARLA simulator	CARLA can provide a resolution of 1.7 MP (1820 x 940).
	N-R1.3_16	CARLA simulator	CARLA can provide a horizontal field of view 110°.
	N-R1.3_17	CARLA simulator	CARLA can provide a vertical field of view 47°.
	N-R1.3_18	CARLA simulator	CARLA can provide video data rate must run at 2x16 fps.

	N-R1.3_19	CARLA simulator	CARLA includes Image Signal Processing (ISP) and camera control.
	N-R1.3_20	CARLA simulator	CARLA can meet this requirement.
	N-R1.3_21	CARLA simulator	CARLA camera model can deliver an output image to a middleware that ensure the transition to other subsystems.
	N-R1.3_22	CARLA simulator	CARLA can meet the set of requirements for the perception DF of the simulation framework.
Subject vehicle / AD function	N-R1.3_23	CARLA simulator	CARLA can meet the set of requirements for the maneuvers in the simulations for the examined scenartios
	N-R1.3_24	CARLA simulator	CARLA camera model can deliver an object list with 2D positions of the objects in an image plane with corresponding BoundingBox and object class
	N-R1.3_25	CARLA simulator	CARLA camera model can deliver an object list containing road boundaries (lane detection function) with their coordinates, type of line and number of lanes
	N-R1.3_26	CARLA simulator	CARLA can meet the set of behaviours for the validation of the ADS in the examined scenarios.
	N-R1.3_27	CARLA simulator	Fulfilled
Subject vehicle / vehicle dynamics	R1.3_06	CARLA simulator	CARLA extracts annotated recorded data or simulated object-level data (including uncertainties) derived from the perception layer of the subject vehicle and other vehicles in the vicinity.
	N-R1.3_28	CARLA simulator	CARLA vehicle model is able to adapt to a specified sensor position
Traffic agents	N-R1.3_29	CARLA simulator	CARLA supports scenario runner being able to perform ispecified manoeuvers on request of the scenario generator
	N-R1.3_30	CARLA simulator	CARLA supports scenario runner being able to perform specified manoeuvers on request.
Connectivity	R1.3_03	CARLA simulator	CARLA supports hybrid ViL tests, where a real connected vehicle is in parallel to a virtual scenario executed on the cloud, through Vehicle-to-Cloud (V2C) communication.
	R1.3_04	CARLA simulator	CARLA includes sensors, such as cameras, LiDARs, etc., which can be placed anywhere, and support simulated CPM data coming from other

			road users in the vicinity of the ego-vehicle
	R1.3_05	CARLA simulator	CARLA includes sensors, such as cameras, LiDARs, etc., which can be placed anywhere, and support simulated CPM data coming from sensor equipped Road-Side Units (RSUs)
	N-R1.3_31	ns3 network simulator	ns3 supports Cooperative Awareness Messages (CAMs)
	N-R1.3_32	ns3 network simulator	ns3 supports Cooperative Awareness Messages (CAMs)
Simulation model validation	N-R1.3_33	CARLA simulator	CARLA supports scenario runner being able to verify that executed simulations correspond to the requests from the test case manager.

4.1.3.3 Limitations

It can be concluded from Table 5 that CARLA and ns-3 are suitable tools for sub-UC1.3 and may be enhanced or used together with other tools to achieve a complete simulation framework and meet all the defined requirements in the deliverable D4.2 (section 8). There are some minor limitations that can be handle with workarounds, by means of overcoming simulation limitations or/and constraints, such as:

a) communications through the ns-3 S/W tool are modelled at the functional level and there is no low-level physical model of the communication link;

b) the SuT in the simulations is not exactly the same that will be tested on proving grounds.

4.2 Traffic jam AD validation (UC 2)

The scope of the **UC ID 2 "Traffic Jam AD validation"** is to validate the automated lane keeping system (ALKS) for SAE L3+ automated vehicles on motorways and motorway-similar roads via the implementation of a combined validation testing, including virtual simulations and physical tests.

This UC is focusing on AD behaviour validation and aims to optimise the workflow from test case generation to model creation and integration, as well as to test execution and assessment.

In the following subsection 4.2.1, the simulation tooling specification will be provided, based on the defined subsystem requirements in the deliverable D4.2, in order to validate the presented CCAM SuT.

4.2.1 Safety assessment & decision making (sub-UC 2.1)

4.2.1.1 Short description

Sub-UC 2.1 deals with safety assessment and decision making of a Traffic Jam AD, and the following is the selection process of a simulation tooling for this sub-UC.

The requirements defined in the deliverables D7.1 and D4.2 and refined in T4.3 are as follows:

- Nine (9) requirement IDs (R2.1_18, R2.1_19, R2.1_20, R2.1_36, R2.1_37, R2.1_44, R2.1_46, R2.1_51, R2.1_52) are related to the “Test case manager” subsystem.

- Three (3) requirement IDs (R2.1_24, R2.1_53_X, R2.1_54_X(group)) are related to the “environment” subsystem.

- Thirty-one (31) requirement IDs are related to the “subject vehicle” subsystem split into three sub-categories as follows:

    - Six (6) requirement IDs (R2.1_08, R2.1_25, R2.1_26, R2.1_28, R2.1_38, R2.1_39) are related to the “sensors” sub-category.

    - Twenty-three (23) requirement IDs (R2.1_01, R2.1_02, R2.1_03, R2.1_04, R2.1_05, R2.1_06, R2.1_07, R2.1_09, R2.1_10, R2.1_11, R2.1_12, R2.1_13, R2.1_14, R2.1_27, R2.1_29, R2.1_30, R2.1_31, R2.1_32, R2.1_33, R2.1_34, R2.1_35, R2.1_40, R2.1_41) are related to the “AD function” sub-category.

    - Two (2) requirement IDs (R2.1_15, R2.1_16) are related to the “vehicle dynamics” sub-category.

- One (1) requirement ID (R2.1_55_X(group)) is related to the “Traffic agents” subsystem.

- Eleven (11) requirement IDs (R2.1_17, R2.1_21, R2.1_22, R2.1_23, R2.1_42, R2.1_43, R2.1_45, R2.1_47, R2.1_48, R2.1_49, R2.1_50) are related to the “Simulation model validation” subsystem.

4.2.1.2 Mapping of subsystem requirements and corresponding S/W features

All the above requirement IDs for the simulation tooling specification of the sub-UC 2.1 are depicted in Table 6 in relation to the subsystems of the simulation framework, the S/W tool names and the S/W tool features, which meet these requirements.

Table 6: Subsystems, requirements and S/W tool features for the simulation tooling specification of the sub-UC 2.1.

Subsystem	Req number ID	S/W tool name	S/W tool feature that meets the requirement
Test case manager	R2.1_18	AVL OpenXOntology	AVL has a tooling based on ASAM OpenXOntology

			standard that is capable of ODD definition. AVL's tooling does not have to be used to follow this approach.
	R2.1_19	AVL ScenarioDesigner AVL ModelCONNECT	ScenarioDesigner --> develop OpenDrive and OpenScenario format scenarios Model.CONNECT --> connect, manage and execute scenarios
	R2.1_20	AVL ScenarioDesigner	AVL provides manoeuvre catalogue for proving ground tests.
	R2.1_36	AVL ScenarioDesigner AVL ModelCONNECT	AVL supports lane keeping scenarios.
	R2.1_37	AVL ScenarioDesigner AVL ModelCONNECT	AVL has smart scenario generation method tools to complement ODDs
	R2.1_44	AVL ModelCONNECT	KPIs defined directly using C and/or python code within. Model.CONNECT with addition of built-in KPIs
	R2.1_46	AVL ModelCONNECT	AVL supports a built-in scenario management tool to allow manual selection and parametrization of test cases.
	R2.1_51	AVL ModelCONNECT	AVL supports easy and quick evaluation and reporting of results.
	R2.1_52	AVL ModelCONNECT	AVL supports configurable and adaptable KPIs and their thresholds.
Environment	R2.1_24	CARLA SIMULATOR	Sensor speicification through CARLA's available sensors (radar, lidar, camera, ultrasonic)
	R2.1_53_X(group)	CARLA SIMULATOR Esmini	Depending on complexity, all these requirements can be satisfied in either CARLA or Esmini or both in parallel simulations
	R2.1_54_X(group)	CARLA SIMULATOR	Possiblity to add environmental effects
Subject vehicle - sensors	R2.1_08	MATLAB SIMULINK FMU	ADS development in Simulink, deployed through FMU
	R2.1_25	CARLA SIMULATOR	Allows to add sensors in the subject vehicle and configure all required characteristics

	R2.1_26	MATLAB SIMULINK FMU	Perception developed in Simulink, deployed through FMU
	R2.1_28	MATLAB SIMULINK FMU	Depends on hardware used, but is possible.
	R2.1_39	CARLA SIMULATOR FMU	Perception algorithm developed to fulfill the requirement
Subject vehicle – AD function	R2.1_01	MATLAB SIMULINK FMU	MATLAB and ADS developed can satisfy this requirement
	R2.1_02	MATLAB SIMULINK FMU	MATLAB and ADS developed can satisfy this requirement
	R2.1_03	MATLAB SIMULINK FMU	MATLAB and ADS developed can satisfy this requirement
	R2.1_04	MATLAB SIMULINK FMU	MATLAB and ADS developed can satisfy this requirement
	R2.1_05	MATLAB SIMULINK FMU	MATLAB and ADS developed can satisfy this requirement
	R2.1_06	MATLAB SIMULINK FMU	MATLAB and ADS developed can satisfy this requirement
	R2.1_07	MATLAB SIMULINK FMU	MATLAB and ADS developed can satisfy this requirement
	R2.1_09	MATLAB SIMULINK FMU	MATLAB and ADS developed can satisfy this requirement
	R2.1_10	MATLAB SIMULINK FMU	MATLAB and ADS developed can satisfy this requirement
	R2.1_11	MATLAB SIMULINK FMU	MATLAB and ADS developed can satisfy this requirement
	R2.1_12	MATLAB SIMULINK FMU	MATLAB and ADS developed can satisfy this requirement
	R2.1_13	MATLAB SIMULINK FMU	MATLAB and ADS developed can satisfy this requirement
	R2.1_14	MATLAB SIMULINK FMU	MATLAB and ADS developed can satisfy this requirement
	R2.1_27	MATLAB SIMULINK FMU	MATLAB and ADS developed can satisfy this requirement
	R2.1_29	MATLAB SIMULINK FMU	MATLAB and ADS developed can satisfy this requirement
	R2.1_30	MATLAB SIMULINK FMU	MATLAB and ADS developed can satisfy this requirement
	R2.1_31	MATLAB SIMULINK FMU	MATLAB and ADS developed can satisfy this requirement
	R2.1_32	MATLAB SIMULINK FMU	MATLAB and ADS developed can satisfy this requirement
	R2.1_33	MATLAB SIMULINK FMU	MATLAB and ADS developed can satisfy this requirement
	R2.1_34	MATLAB SIMULINK FMU	MATLAB and ADS developed can satisfy this requirement

	R2.1_35	AVL ScenarioDesigner AVL Model.Connect	Developed to satisfy the requirement
	R2.1_40	C++ or Python (with CARLA API)	AD functions can be implemented with full freedom in a series of C++ or Python codes to then provide the necessary control commands to the CARLA simulation through its Python API
	R2.1_41	MATLAB SIMULINK FMU	Developed to satisfy the requirement
Subject vehicle – vehicle dynamics	R2.1_15	AVL VSM	AVL VSM includes basic vehicle parameters
	R2.1_16	AVL VSM	Fulfilled. Paired with CARLA it can provide advanced vehicle dynamic calculations investigating complex behaviours like vehicle pitching or other advanced phenomena.
Traffic agents	R2.1_55_X(group)	AVL ScenarioDesigner CARLA SIMULATOR	Traffic agents from CARLA SIMULATOR defined by openScenario from ScenarioDesigner.
Simulation model validation	R2.1_17	CARLA SIMULATOR	Vehicle model development and importing to Carla
	R2.1_21	CARLA SIMULATOR	OpenDrive created and imported to Carla
	R2.1_22	CARLA SIMULATOR	OpenDrive created and imported to Carla
	R2.1_23	CARLA SIMULATOR	CARLA supports this requirement
	R2.1_42	AVL ModelCONNECT	CARLA supports this requirement
	R2.1_43	AVL ModelCONNECT	CARLA supports this requirement
	R2.1_45	AVL ModelCONNECT	CARLA supports this requirement
	R2.1_47	AVL ModelCONNECT	CARLA supports this requirement
	R2.1_48	AVL ModelCONNECT	Supports CAN, FMU, ROS, Python and may more.
	R2.1_49	AVL ModelCONNECT	Fulfilled
	R2.1_50	AVL ModelCONNECT	Model.CONNECT is proven to be deterministic and repeatable

4.2.1.3 Limitations

As mentioned previously, the main goal of the sub-UC 2.1 is to enable virtual safety assessment and decision-making evaluation. The central tool of the simulation tooling for the sub-UC 2.1 is AVL Model.CONNECT that acts as a binding glue for other specialized software and ensures repeatability.

Many different tools can be used for other model sub-systems but for this sub-UC CARLA is suggested as the simulation environment along with the Esmini scenario player.

Esmini is not necessary if the environment software supports OpenDRIVE/OpenSCENARIO but in the present sub-UC Esmini is used to run native OpenX standard. Test scenarios with OpenX standards are developed and fully parametrized with AVL ScenarioDesigner.

Vehicle model is developed in CARLA and in AVL VSM, as VSM is the tool used for advanced vehicle dynamics simulation that communicated with CARLA and Esmini in order to coordinate all vehicle sub-systems in a cohesive co-simulation.

Different sensors can be modelled with CARLA as well. VSM can be used to model the impact of vehicle dynamics on sensor perception performance and the resulting impact on the systems decision making.

The ALKS SuT will be integrated into Model.CONNECT via Simulink block or an FMU generated from Simulink development environment. And finally, Model.CONNECT will serve as a test-case manager, simulation variation and model validation platform.

Following the above, in parallel with the analysis in Table 6, it could be stated that the above selected simulation tools cover all the requirements of the present sub-UC and no limitations can be found.

4.3 Highway (co-operative) AD validation (UC 3)

The scope of **UC 3 - Highway (co-operative) AD validation** is to validate semi/highly automated vehicles (SAE L2/L3+) on motorways (and similar roads) via the implementation of a hybrid validation approach (virtual simulations and physical tests).

UC 3 - Highway (co-operative) AD validation includes two main sub-UCs as follows:

- **sub-UC 3.1 - map-based perception & decision-making & control testing:** focuses on demonstrating how the vehicle's safety and awareness can be improved based on information coming from maps, sensors or connected services about road characteristics or road dynamic events.

- **sub-UC 3.2 - cooperative perception & decision making & control testing:** focuses on demonstrating how safety and surrounding awareness can be improved on motorways by including cooperative V2X functionality (with other vehicles in the

neighbourhood) in the Highway Pilot (HWP) system (e.g., by leveraging and upgrading the driver assistance functionality developed previously in C-ACC from sub-UC 1.2).

In the following subsections 3.3.1 and 3.3.2, simulation tooling specifications will be provided, based on the defined subsystem requirements in D4.2, in order to validate the advanced C-ACC functionality presented in the above sub-UCs.

4.3.1 Map based perception & decision making (sub-UC 3.1)

4.3.1.1 Short description

In sub-UC 3.1 SUNRISE SAF is applied to a map-based AD functions in highway scenarios. The SuT will be an advanced ACC/HWP that is expected to control the longitudinal dynamics based on High Definition (HD) map data, sensors and V2X connectivity.

For the present sub-UC, three functional scenarios are defined in deliverable D7.1: adapting speed to varying speed limits and adapting speed to varying road curvature (with information from sensors/maps/V2I). A third functional scenario is considering green driving on slopes.

With respect to the simulations for sub-UC 3.1, these will run in the IPG CarMaker environment with a model of the SuT. Map data is provided to the self-driving agent by the host CarMaker environment. Speed limit signs can also be encoded. V2X communications are simulated via SimNet (a CarMaker add-on that permits to run multiagent simulations).

Based on the work in the deliverable D4.2 (chapter 5), ten (10) specific requirements were identified in total for sub-UC 3.1, based on the relevant subsystems of the SUNRISE Harmonized V&V Simulation Framework. These requirements are a refined version of the ones defined in the deliverable D7.1 and are presented below:

- Three (3) requirement IDs (R3.1_01, R3.1_06_X(group), R3.1_10) are related to the “test case manager” subsystem.
- Two (2) requirement IDs (R3.1_02_X(group), R3.1_03_X(group)) are related to the “environment” subsystem.
- Four (4) requirement IDs (R3.1_05, R3.1_06_01, R3.1_06_02, R3.1_07) are related to the “subject vehicle” subsystem.
- One (1) requirement ID (R3.1_04_X(group)) is related to the “traffic agents” subsystem.
- One (1) requirement ID (R3.1_10) is related to the “simulation model validation” subsystem.

4.3.1.2 Mapping of subsystem requirements and corresponding S/W features

Table 7 lists the requirements for the simulation tools from D4.2, and how they are met by the simulation systems of the involved partners.

Table 7: Subsystems, requirements and S/W tool features for the simulation tooling specification of the sub-UC 3.1.

Subsystem	Req number ID	S/W tool name	S/W tool feature that meets the requirement
Test case manager	R3.1_01	IPG CarMaker with SIMNET	In principle any simulation parameter and state is measurable (the exact KPI should be specified).
	R3.1_06_X(group)	IPG CarMaker with SIMNET	The subject vehile must comply with the speed limit when entering the speed limit section. It must also adapt the speed in a curve to meet (safe) human-like behaviour [47].
	R3.1_10	IPG CarMaker with SIMNET	A possible issue is that the system available in CarMaker is no longer the exact system as implemented in the test vehicles, and thus, may be a mismatch between the physical and virtual SuT. Another aspect to be considered is the road curvature profile. A simple straight followed by a curve may not be enough because the SuT must be tested against random plausible variations in the curvature. Also the SuT should be tested against varying map horizons.
Environment	R3.1_02_X	IPG CarMaker with SIMNET	The simulation environment in CarMaker allows modeling these features, albeit some may not be immediate. Not all will be tested.
	R3.1_03_X	IPG CarMaker with SIMNET	Wind can be modelled. The influence of rain can be modeled by varying friction coefficient. The effect of atmospheric conditions on V2V and sensors will not be studied.
Subject vehicle / sensors	R3.1_05	IPG CarMaker with SIMNET and three different agents.	There are three different self-driving agents; co-driver, motion planner and simple controller based in the IDM. The former two supports all types of longitudinal and lateral manoeuvers (IDM only longitudinal ones). The system's input is made of both an electronic horizon and signals that represent high-level lane camera detection with information about the lane confidence. As for other objects the system input may be a list of (up to 20) obstacles in any relative position and state. The system makes

			inference of object intentions for their prediction if the intentions and future paths are not received via V2V. If obstacles are on roads that are not in the map, the prediction is degraded. Cut-in manoeuvers are predicted with high precision. The system's plans optimal efficient collision free manoeuvers choosing from a large number of alternatives speed limits coming from the digital maps. The co-driver is an agent that can collaborate with a human driver (but this is not tested in this sub-UC). It can run in shadow mode.
	R3.1_07	IPG CarMaker with SIMNET and three different agents.	Speed limits and curvature profiles msut be in the maps. General safer behaviour in adverse weather conditions may be obtained via setting a reduced requested cruising speed (this on top of self-driving agent functions). In the simulations, it has been assumed that the SuT is not aware of its ODD, and hence, a crossing of ODD limits is not detected.
Subject vehicle / AD function	R3.1_06_01	IPG CarMaker with SIMNET and three different agents.	The three agents adapt (in principle) to any speed limit below the requested speed. No interaction between SuT and the human driver can be tested and is planned in simulation.
	R3.1_06_02	IPG CarMaker with SIMNET and three different agents.	Two agents adapt (in principle) to any curvature profile, producing a human-like speed choice in curves. No interaction between SuT and the human driver can be tested and is planned in simulation.
Traffic agents	R3.1_04_X(group)	IPG CarMaker with SIMNET and three different agents.	VRU can be modelled. There also is a safe speed neural function for recommendi safe speed in case of distrated pedestrians.
Simulation model validation	R3.1_10	IPG CarMaker with SIMNET	A possible issue is that the system available in CarMaker is no longer the exact system as implemented in the test vehicles, and thus, may be a mismatch between the physical and virtual SuT. Another aspect to be considered is the road curvature profile. A simple straight followed by a curve may not be enough because the SuT must be tested against random plausible variations in the

			curvature. Also the SuT should be tested against varying map horizons.

4.3.1.3 Limitations

It can be concluded from Table 7 that CarMaker is a suitable tool for sub-UC3.1 and may be enhanced or used together with other tools to achieve a complete simulation framework and meet all the defined requirements in the deliverable D4.2 (section 8). There are some minor limitations that can be handle with workarounds, by means of overcoming simulation limitations or/and constraints, such as:

a) KPIs should be specified for the present subUC 3.1 to be sure that can be met by the CarMaker tool;

b) communications are modelled at the functional level and there is no low-level physical model of the communication link;

c) It has been assumed that the SuT is not aware of its ODD, and hence, a crossing of ODD limits is not detected;

d) the SuT in the simulations is not exactly the same that will be tested on proving grounds;

e) a dynamic speed limit is currently unsupported.

4.3.2 Cooperative perception & decision making & control (sub-UC 3.2)

4.3.2.1 Short description

In the sub-UC 3.2 SUNRISE SAF is applied to the cooperative AD functions in highway scenarios. The SuT is an advanced ACC with V2X communication like CAM coming from other vehicles that are expected to control the longitudinal dynamics.

For the present sub-UC, four functional scenarios are defined in deliverable D7.1: a) ACC adapting speed to leading vehicles, b) ACC adapting to decelerating vehicles, c) ACC anticipating cutting-in cooperative vehicles, d) ACC reacting to the loss of control of a leading vehicle.

With respect to the simulations for sub-UC 3.1, these will run in the IPG CarMaker environment with a model of the SuT. V2V communication is simulated via SimNet (a CarMaker add-on that permits to run multiagent simulations).

Based on the work in the deliverable D4.2 (chapter 5), thirteen (13) specific requirements were identified in total for sub-UC 3.2, based on the relevant subsystems of the SUNRISE Harmonized V&V Simulation Framework. These requirements are a refined version of the ones defined in the deliverable D7.1 and are presented below:

- One (1) requirement ID (R3.2_10) is related to the “test case manager” subsystem.
- Two (2) requirement IDs (R3.1_02_X(group), R3.1_03_X(group)) are related to the “environment” subsystem.
- Five (5) requirement IDs (R3.2_05, R3.2_06_01, R3.2_06_02, R3.2_06_03, R3.2_07) are related to the “subject vehicle” subsystem.
- Two (2) requirement IDs (R3.2_04_01, R3.2_07) are related to the “traffic agents” subsystem.
- Two (2) requirement IDs (R3.2_04_02 and R3.2_08) are related to the “connectivity” subsystem
- One (1) requirement ID (R3.2_10) is related to the “simulation model validation” subsystem.

4.3.2.2 Mapping of subsystem requirements and corresponding S/W features

Table 8 lists the requirements for the simulation tools from D4.2, and how they are met by the simulation systems of the involved partners.

Table 8: Subsystems, requirements and S/W tool features for the simulation tooling specification of the sub-UC 3.2.

Subsystem	Req number ID	S/W tool name	S/W tool feature that meets the requirement
Test case manager	R3.2_10	IPG CarMaker with SIMNET	A possible issue is that the system available in CarMaker is no longer the exact system as implemented in the test vehicles, and thus, may be a mismatch between the physical and virtual SuT. Another aspect to be considered is the road curvature profile. A simple straight followed by a curve may not be enough because the SuT must be tested against random plausible variations in the curvature. Also the SuT should be tested against varying map horizons.
Environment	R3.2_02_X	IPG CarMaker with SIMNET	The simulation environment in CarMaker allows modeling these features, albeit some may not be immediate. Not all will be tested.
	R3.2_03_X	IPG CarMaker with SIMNET	Wind can be modelled. The influence of rain can be modeled by varying friction coefficient. The effect of atmospheric conditions on V2V and sensors will not be studied.

Subject vehicle / sensors	R3.2_07	IPG CarMaker with SIMNET and three different agents.	Speed limits and curvature profiles must be in the maps. General safer behaviour in adverse weather conditions may be obtained via setting a reduced requested cruising speed (this on top of self-driving agent functions). In the simulations, it has been assumed that the SuT is not aware of its ODD, and hence, a crossing of ODD limits is not detected.
Subject vehicle / AD function	R3.2_05	IPG CarMaker with SIMNET and three different agents.	There are three different self-driving agents; co-driver, motion planner and simple controller based in the IDM. The former two supports all types of longitudinal and lateral manoeuvers (IDM only longitudinal ones). The system's input is made of both an electronic horizon and signals that represent high-level lane camera detection with information about the lane confidence. As for other objects the system input may be a list of (up to 20) obstacles in any relative position and state. The system makes inference of object intentions for their prediction if the intentions and future paths are not received via V2V. If obstacles are on roads that are not in the map, the prediction is degraded. Cut-in manoeuvers are predicted with high precision. The system's plans optimal efficient collision free manoeuvers choosing from a large number of alternatives speed limits coming from the digital maps. The co-driver is an agent that can collaborate with a human driver (but this is not tested in this sub-UC). It can run in shadow mode.
	R3.2_06_01	IPG CarMaker with SIMNET and three different agents	Scenarios in the present sub-UC can be simulated.
	R3.2_06_02	IPG CarMaker with SIMNET and three different agents	Scenarios in the present sub-UC can be simulated.
	R3.2_06_03	IPG CarMaker with SIMNET and three different agents	Scenarios in the present sub-UC can be simulated.

Subject vehicle / vehicle dynamics	R3.2_05	IPG CarMaker	Vehicle dynamics models in CarMaker are state of the art models (in particular with 14 degrees of freedom).
Traffic agents	R3.2_04_01	IPG CarMaker with SIMNET and three different sagents	VRU can be modelled. There also is a safe speed neural function for recommend safe speed in case of distracted pedestrians.
	R3.2_07	IPG CarMaker with SIMNET and three different agents	Speed limits and curvature profiles must be in the maps. General safer behaviour in adverse weather conditions may be obtained via setting a reduced requested cruising speed (this on top of self-driving agent functions). In the simulations, it has been assumed that the SuT is not aware of its ODD, and hence, a crossing of ODD limits is not detected.
Connectivity	R3.2_04_02	IPG CarMaker with SIMNET, V2V emulation and three different self-driving agents	Communication is emulated with a software layer connecting the SIMNET threads.
	R3.2_08	IPG CarMaker with SIMNET, V2Ω emulation and three different self-driving agents	Communication is emulated with a software layer connecting the SIMNET threads.
Simulation model validation	R3.2_10	IPG CarMaker	A possible issue is that the system available in CarMaker is no longer the exact system as implemented in the test vehicles, and thus, may be a mismatch between the physical and virtual SuT. Another aspect to be considered is the road curvature profile. A simple straight followed by a curve may not be enough because the SuT must be tested against random plausible variations in the curvature. Also the SuT should be tested against varying map horizons.

4.3.2.3 Limitations

It can be concluded from Table 8 that CarMaker is a suitable tool for sub-UC3.2 and may be enhanced or used together with other tools to achieve a complete simulation framework and meet all the defined requirements in the deliverable D4.2 (section 8). There are some minor

limitations that can be handle with workarounds, by means of overcoming simulation limitations or/and constraints, such as:

a) KPIs should be specified for the present subUC 3.2 to be sure that can be met by the CarMaker tool;

b) communications are modelled at the functional level and there is no low-level physical model of the communication link;

c) It has been assumed that the SuT is not aware of its ODD, and hence, a crossing of ODD limits is not detected;

d) the SuT in the simulations is not exactly the same that will be tested on proving grounds;

4.4 Freight vehicle automated parking validation (UC 4)

The scope of **"UC ID 4 – Freight vehicle automated parking validation"** is to validate the environment perception and connected cyber-security perception for highly automated freight transport vehicles in confined areas via the implementation of a hybrid validation testing, by combining virtual simulations and physical tests. In SUNRISE project, UC ID 4 includes two main sub-UCs as follows:

- Sub-UC 4.1: Testing the perception & decision making of the SuT in a truck at low speed.

- Sub-UC 4.2: Testing the connected perception cyber-security of the SuT in a truck at low speed.

In both cases, starting from a pre-defined area, the truck will reverse into a loading dock. A sensor mounted on the loading dock will monitor the area behind the truck and communicate its observations to the truck.

In the following subsections 4.4.1 and 4.4.2, simulation tooling specifications will be provided, based on the defined subsystem requirements in D4.2, in order to validate the SuT (automated reverse function) presented in the above sub-UCs.

4.4.1 Truck low-speed perception & decision making (sub-UC 4.1)

4.4.1.1 Short description

The following analysis relates to the specification of a simulation tooling for the sub-UC 4.1, through the SUNRISE SAF, for validation the safety of the lowspeed reversing of a truck with trailer system, within a predefined ODD context. The truck-trailer system communicates with a logistic terminal to get data about any objects obstructing the reversing path.

Based on the work in the deliverable D4.2 (chapter 5), sixteen (16) specific requirements were identified in total for sub-UC 4.1, based on the relevant subsystems of the SUNRISE

Harmonized V&V Simulation Framework. These requirements are a refined version of the ones defined in the deliverable D7.1 and are presented below:

- Two (2) requirement IDs (R4.1_12, R4.1_13) are related to the “test case manager” subsystem.
- Three (3) requirement IDs (R4.1_05_X(group), R4.1_06_X(group), R4.1_07) are related to the “environment” subsystem.
- Nine (9) requirement IDs (R4.1_01_00, R4.1_01_01, R4.1_01_02, R4.1_02, R4.1_03, R4.1_04, R4.1_14, R4.1_08, R4.1_09) are related to the “subject vehicle” subsystem.
- One (1) requirement ID (R4.1_15) is related to the "connectivity" subsystem.
- One (1) requirement ID (R4.1_16) is related to the simulation model validation

In addition to these, in this task two requirements with IDs (R4.1_01_04, R4.1_01_05) were identified as requirements for the logistic hub.

4.4.1.2 Mapping of subsystem requirements and corresponding S/W features

Table 9 lists the requirements for the simulation tools from D4.2, and how they are met by the simulation systems of the involved partners.

Table 9: Subsystems, requirements and S/W tool features for the simulation tooling specification of the sub-UC 4.1.

Subsystem	Req number ID	S/W tool name	S/W tool feature that meets the requirement
Test case manager	R4.1_12	WayWiseR	Custom-build to work with WayWiseR and handles the initial scenario provided
	R4.1_13	WayWiseR	Tool developed specifically for managing the testcases and validating them against real truck data.
Environment	R4.1_05_X(group)	CARLA SIMULATOR	CARLA supports these requirements
	R4.1_06_X(group)	CARLA SIMULATOR	CARLA supports these requirements
	R4.1_07	CARLA SIMULATOR	CARLA supports this requirement
Subject vehicle	R4.1_01_00	WayWiseR	WayWiseR contains object detection algorithms.
	R4.1_01_01	CARLA SIMULATOR	CARLA can simulate positioning sensors
	R4.1_01_02	WayWiseR	Perception sensor fusing algorithms exist in WayWiseR that work on the data coming from CARLA

	R4.1_02	WayWiseR	Ability to plan and create routes for backing
	R4.1_03	WayWise	Contains the backing function
	R4.1_04	WayWise	Contains the backing function
	R4.1_14	CARLA SIMULATOR	Ability to simulate the required truck.
	R4.1_08	CARLA SIMULATOR	CARLA supports this requirmenet
	R4.1_09	CARLA SIMULATOR+WayWiseR	CARLA and WayWiseR supports this requirmenet
Connectivity	R4.1_15	WayWiseR	Built on ROS2 and thus provinds connectivity in built.
Simulation model validation	R4.1_16	WayWiseR	Tool developed specifically for managing the testcases and validating them against real truck data.

4.4.1.3 Limitations

The focus of this sub-UC is to evaluate the SAF and demonstrate its benefits using the simulation tooling, with CARLA being the central tool providing the simulation environment. A truck is modeled in CARLA to represent the physical truck used in this sub-UC. Additionally, CARLA helps model the truck's sensors, including GNSS, Camera, and Angle sensors.

The SuT (reversing function) is implemented in WayWise, which provides a common implementation of the backing function for both the simulation and the physical truck setup. WayWiseR, a ROS2-based interface to WayWise, is used to connect the truck in CARLA with the rest of the simulation system. Furthermore, as a rapid prototyping platform, WayWiseR incorporates the test case manager and simulation model validation.

It can be concluded from Table 9 that the presented simulation tools are suitable for sub-UC4.1 and may be enhanced or used together with other tools to achieve a complete simulation framework and meet all the defined requirements in the deliverable D4.2 (section 8). There are some minor limitations that can be handle with workarounds, by means of overcoming simulation limitations or/and constraints, such as:

a) KPIs should be specified for the present subUC 4.1 to be sure that can be met by the CARLA and WayWiseR simulation tools;

b) communications are modelled at the functional level and there is no low-level physical model of the communication link;

c) the SuT in the simulations is not exactly the same that will be tested on proving grounds;

4.4.2 Truck low-speed connected perception cyber-security testing (sub-UC 4.2)

4.4.2.1 Short description

The following analysis relates to the specification of a simulation tooling for the sub-UC 4.2, in order to validate the truck low-speed connected perception cyber-security testing, within a predefined ODD context. In this sub-UC, the testing and validation activities are focused on cybersecurity by focusing on the connection between the low-speed truck and the docking area.

Based on the work in the deliverable D4.2 (chapter 8, table 10), sixteen (16) specific requirements were identified in total for sub-UC 4.2, based on the relevant subsystems of the SUNRISE Harmonized V&V Simulation Framework. These requirements are a refined version of the ones defined in the deliverable D7.1. Moreover, after refining the test objectives for sub-UC 4.2 during the activities of the present task, additional requirements were identified, besides the ones defined in the deliverables D7.1 and D4.2. These additional requirements are denoted as “N-”, their descriptions can be found in annex 2.

- Four (4) requirement IDs (N-R4.2_13, N-R4.2_14, N-R4.2_15, N-R4.2_16) are related to the “test case manager” subsystem.

- Five (5) requirement IDs (R4.2_05, R4.2_06, R4.2_07, N-R4.2_17, N-R4.2_18) are related to the “environment” subsystem.

- Fifteen (15) requirement IDs (R4.2_08, R4.2_09, R4.2_10, R4.2_11, R4.2_12, N-R4.2_19, N-R4.2_20, N-R4.2_21, N-R4.2_22, N-R4.2_23, N-R4.2_24, N-R4.2_25, N-R4.2_26, N-R4.2_27, N-R4.2_28) are related to the “subject vehicle” subsystem.

- Two (2) requirement IDs (N-R4.2_29, N-R4.2_30) are related to the “traffic agents” subsystem.

- Two (2) requirement IDs (N-R4.2_31, N-R4.2_32) are related to the “connectivity” subsystem.

- Five (5) requirement IDs (R4.2_01, R4.2_02, R4.2_03, R4.2_04, N-R4.2_33) are related to the “simulation model validation” subsystem.

4.4.2.2 Mapping of subsystem requirements and corresponding S/W features

All the above requirement IDs for the simulation tooling specification of the sub-UC 4.2 are depicted in Table 10 in relation to the subsystems of the simulation framework, the S/W tool names and the S/W tool features, which meet these requirements.

Table 10: Subsystems, requirements and S/W tool features for the simulation tooling specification of the sub-UC 4.2.

Subsystem	Req number ID	S/W tool name	S/W tool feature that meets the requirement
Test case manager	N-R4.2_13	CARLA simulator	CARLA supports OpenScenraio format which reads as input a scenario description file and generates a corresponding scenario environment.
	N-R4.2_14	CARLA simulator	CARLA has a configuration interface allowing the user to configure at least the sensor position and its basic parameters.
	N-R4.2_15	CARLA simulator	CARLA can store at least 1 hour of video data from the camera.
	N-R4.2_16	CARLA simulator	CARLA has a dedicated module in analysing the scenario results and generating metrics for the test coverage evaluation.
Environment	N-R4.2_17	CARLA simulator	CARLA can visually represent a connected perception cyber-security testing environment.
	R4.2_05	CARLA simulator	CARLA supports the ODD/Scenery elements for all the examined test scenarios.
	N-R4.2_18	CARLA simulator	CARLA supports Software-in-the-Loop (SiL), Model-in-the-Loop (MiL) and CoSim methods.
	R4.2_06	CARLA simulator	CARLA supports the ODD/Atmospheric conditions elements for all the examined test scenarios.
	R4.2_07	CARLA simulator	CARLA supports the ODD/Dynamic elements (traffic participants) behaviour elements for all the examined test scenarios.
Subject vehicle / sensors	N-R4.2_19	CARLA simulator	CARLA can provide a resolution of 1.7 MP (1820 x 940).
	N-R4.2_20	CARLA simulator	CARLA can provide a horizontal field of view 110°.
	N-R4.2_21	CARLA simulator	CARLA can provide a vertical field of view 47°.
	N-R4.2_22	CARLA simulator	CARLA can provide video data rate must run at 2x16 fps.
	N-R4.2_23	CARLA simulator	CARLA include ISP and camera control.
	N-R4.2_24	CARLA simulator	CARLA can meet this requirement.

	N-R4.2_25	CARLA simulator	CARLA camera model can deliver an output image to a middleware that ensure the transition to other subsystems.
	N-R4.2_26	CARLA simulator	CARLA can meet the set of requirements for the perception DF of the simulation framework.
Subject vehicle / AD function	R4.2_08	CARLA simulator	CARLA can meet this requirement.
	R4.2_09	CARLA simulator	CARLA can meet this requirement.
	R4.2_10	CARLA simulator	CARLA can meet this requirement.
	R4.2_11	CARLA simulator	CARLA can meet this requirement.
	R4.2_12	CARLA simulator	CARLA can meet this requirement.
	N-R4.2_27	CARLA simulator	CARLA can meet this requirement.
Subject vehicle / vehicle dynamics	N-R4.2_28	through	CARLA vehicle model is able to adapt to a specified sensor position via a Python API
Traffic agents	N-R4.2_29	CARLA simulator	CARLA supports scenario runner being able to perform specified manoeuvers on request of the scenario generator
	N-R4.2_30	CARLA simulator	CARLA supports scenario runner being able to perform specified manoeuvers on request.
Connectivity	N-R4.2_31	ns3 network simulator	ns3 supports Cooperative Awareness Messages (CAMs)
	N-R4.2_32	ns3 network simulator	ns3 supports Cooperative Awareness Messages (CAMs)
Simulation model validation	R4.2_01	CARLA simulator	CARLA can meet this requirement.
	R4.2_02	CARLA simulator	CARLA can meet this requirement.
	R4.2_03	CARLA simulator	CARLA can meet this requirement.
	R4.2_04	CARLA simulator	CARLA can meet this requirement.
	N-R4.2_33	CARLA simulator	CARLA supports scenario runner being able to verify that executed simulations correspond to the requests from the test case manager.

4.4.2.3 Limitations

The focus of this sub-UC is to evaluate the SAF and demonstrate its benefits using the simulation tooling, with CARLA being the central tool providing the simulation environment. A truck is modeled in CARLA to represent the physical truck used in this sub-UC. Additionally, CARLA helps model the truck's sensors, including GNSS, Camera, and Angle sensors. The ns3 network simulator is used for simulating the connectivity features of this sub-UC.

It can be concluded from Table 10 that the presented simulation tools (CARLA and ns-3) are suitable for sub-UC4.2 and may be enhanced or used together with other tools to achieve a complete simulation framework and meet all the defined requirements in the deliverable D4.2 (section 8). There are some minor limitations that can be handle with workarounds, by means of overcoming simulation limitations or/and constraints, such as:

a) KPIs should be specified for the present subUC 4.1 to be sure that can be met by the CARLA and WayWiseR simulation tools;

b) communications are modelled at the functional level and there is no low-level physical model of the communication link;

c) the SuT in the simulations is not exactly the same that will be tested on proving grounds;

4.5 Generic requirements

The generic requirements identified in T4.2 are based on AD safety standards and thus are applicable to all UCs and sub-UCs in the SUNRISE project. The approach towards identifying suitable tools for these general requirements was to perform a *broad survey of tools* instead of a deep investigation into any specific tools. This was necessary as the UC-specific tools differ among the UCs and incorporating other tools specifically to meet the general requirements would create the need for multiple tools per simulation subsystem. Therefore, from a practical point-of-view, it was more interesting to do a broad survey of tools for these requirements. It was observed during the initial investigation that for several requirements, no tools were found which could meet the requirements directly or it was unclear if the requirements could be met. Therefore, we included within the survey supporting tools and methodologies, which could be built upon, or used to enhance existing tools to better meet the requirements.

Before presenting the results of the performed survey, few observations are made about the general requirements identified in the deliverable D4.2:

1. Ten general requirements related to SOTIF standard are for subsystems external to the simulation framework and relate to topics like scenario selection, safety assessment of the SuT, etc. Thus, tools and methodologies for such external requirements are outside the scope of task T4.3 and are part of other SUNRISE deliverables. The requirement IDs for these requirements are: *R10.1.19.2, R10.1.20.2, R10.1.21.7, R10.1.6, R10.1.17.2, R10.1.19.3, R10.1.20.3, R10.1.21.8, R10.1.18.2 and R10.1.15.2.*

2. Requirement ID *R10.1.21.3* related to SOTIF standard relates to AD behavior specification and therefore is not mapped to simulation tools.

3. Seven general requirements related to the SOTIF standard are workflow and/or interfacing requirements for the simulation framework, and not tooling requirements.

Such requirements require workflows and interfaces to be defined, not tools. Table 11 presents these requirements, with recommendations on how they could be addressed in the SAF.

Table 11: Subsystems generic requirements, and recommendation on how to address workflow and interfacing requirements for SAF.

Subsystem	Req number ID	Recommendations on how it could be addressed in SAF
Test case manager	R10.1.16.1	Workflow within simulation framework
	R10.1.17.1	• Workflow within simulation framework. • Interface to analysis blocks of SAF.
	R10.1.18.1	• Scenario diversity metric may be used from T5.3 • Interface to data framework of SAF.
Simulation model validation	R10.1.16.2	Methodology defined in T4.5
	R10.1.19.1	Methodology defined in T4.5
	R10.1.20.1	Methodology defined in T4.5
	R10.1.21.6	Methodology defined in T4.5

4. Tools for four general requirements related to the SOTIF standard cannot be determined independent of evaluating the requirements individually for each sub-UC. The SOTIF standard specifies testing of the SuT at the ODD boundary and outside the ODD, to ensure that the SuT is safe at the ODD boundary but also can perform a safety manoeuver / hand back control safely outside the ODD. Such testing imposes requirements to support such outside ODD conditions on the environment, sensor, vehicle dynamics, and traffic agents subsystems of the simulation tooling. Since the ODD boundary and subsequently the outside ODD conditions are specific to each sub-UC, no conclusions for suitable tools can be drawn without assessing them per sub-UC. The requirement IDs for these requirements are: *R10.1.21.1, R10.1.21.2, R10.1.21.4, R10.1.21.5.*

Table 12 presents the outcome of the survey of tools as well as supporting tools and methodologies. Note that all the requirements in the table pertain to the test case manager subsystem. There are six requirements for which no tools could be identified which could meet the specified requirements. An important reason for this is the limited number of tools and products in industry which address the requirements of the SOTIF standard. However, details on existing tools which would help evaluate if certain requirements could be met is also limited online due to the commercial nature of the product. For example, it is difficult to evaluate R10.1.14, *sufficient justification (safety argumentation) shall be provided for the metrics used to identify unknown-unsafe scenarios*, as there is insufficient information available for tools which claim solutions to identify unknown-unsafe scenarios (e.g., Foretellix Foretify tool).

Many supporting methodologies are found during our survey of the tools. A few are standards which help specify clear machine-readable formats. The majority are research papers, providing methods and metrics for search space definition, scenario exploration, and coverage evaluation. These methods could be implemented to complement existing tools to help better address the requirements of the SOTIF standard.

Table 12: Subsystems, generic requirements and S/W tool features for the simulation tooling specification. All requirements are for subsystem: test case manager.

Req number ID	S/W tool name	S/W tool feature that meets the requirement	Supporting tools and methodologies
R10.1.7	No known tools	n/a	Standards & literature to establish a well-defined machine-readable definition of ODD: • OpenODD standard [48] provides a machine-readable format which could be used to define a search space. However, OpenODD is not yet available. • ISO 34503 [49] provides a taxonomy of ODD attributes. • Irvine et al [50]
R10.1.8	Simcenter Prescan	Scene editor, which creates a search space based on real-world trajectory data.	Search space from real-world data in literature: • Nakamura et al. [51] • Gelder et al. [52]
	Fortellix Foretify	Randomization of parameters of observed scenarios	
	NVIDIA drive labs	STRIVE, optimization over observed scenarios to find critical scenarios.	
R10.1.9	No known tools	n/a	Coverage evaluation methods in literature: • Laurent et al. [53] • Hauer et al. [54] • Hungar [55] • Amersbach et al. [56]
R10.1.10	Simcenter HEEDS	Each tool provides techniques such as optimization, smart sampling, and randomization for effective exploration of the search space within a finite number of runs.	Exploration methods in literature: • Optimization –Klischat et al. [57] • Combinatorial – Rocklage et al. [58] • Sampling-based – Medrano-Berumen et al. [59]
	Model.Connect		
	modeFrontier		
	Foretellix Foretify		
R10.1.11	No known tools	n/a	
R10.1.12	No known tools	n/a	

R10.1.13	Siemens Questa	General-purpose coverage evaluation and coverage automation	Same as for R10.1.9.
	Foretellix Foretify	Coverage-driven verification	
R10.1.14	No known tools	n/a	
R10.1.15.1	No known tools	n/a	

As summary,

1. Twenty-two general requirements were excluded when identifying suitable simulation tools due to reasons such as being assigned to subsystems external to the simulation framework, being related to the AD test function, being workflow or interfacing requirements, or UC-specific considerations which requires them to be analyzed per sub-UC. Recommendations are made for some requirements with regards to how they could be addressed in the SAF.

2. A broad survey of tools, supporting tools, and methodologies was performed for the remaining requirements. The tools identified in the survey, for example the Siemens or Fortellix tools, support basic aspects of the SOTIF standard and may be used in the simulation framework. The supporting tools and methodologies can be used to enhance these existing tools, or implement new ones, for requirements where no suitable tools are found. There are six such requirements.

5 GAP ANALYSIS

This section aims to analyze the gaps/limitations of the selected simulation tools, based on the results from the presented simulation tooling specifications for all the defined sub-UCs and subsystems requirements in Chapter 3. In general, from the analysis in the Tables in Chapter 3, the presented simulation tools meet the defined requirements to a great extent. The main limitations are mostly related to:

a) communications which are modelled at the functional level and there is no low-level physical model of the communication link;

b) limited number of tools and products in industry which address the requirements of the SOTIF standard;

Additionally, 22 generic requirements were excluded when identifying suitable simulation tools due to reasons such as being assigned to subsystems external to simulation framework, relating to the SuT, being workflow or interfacing requirements, or sub-UC specific considerations which requires them to be analysed per sub-UC. Recommendations are made for some requirements with regards to how they could be addressed in the SUNRISE SAF.

In the following two sections, CarMaker, SimCenter Prescan and HEEDs simulation tools are analyzed by identifying limitations and subsystems requirements. The way to overcome these limitations/gaps will be done in detail in other tasks within the SUNRISE project that depend on the deliverable D4.3, i.e. tasks T4.4, T7.2 and T7.3.

5.1 CarMaker

By testing a given SuT in SUNRISE scenarios, the SuT must be “glued” to the CarMaker system. This can be efficiently done with custom code that collects sensory and V2X information, and send them to the SuT agent. The SuT returns motor command that control the subject vehicle (which is in place).

The main limitation of the current glue implementation is using high-level sensor models (i.e., object lists) and high-level emulation of communication; i.e., the physical mechanism for signal propagations and sensor working are not simulated in detail.

CarMaker is used in sub-UCs 1.1, 1.2, 3.1 & 3.2. For sub-UC 1.1, Carmaker is coupled to an external sensor model. In the other three sub-UCs, sensor and communication malfunctions (if required among the logical scenario parameters) have to be summarized with their high-level effect (e.g., delays or reduced detection range) on the signal input to the SuT. Logical scenarios, to be treatable, must in any case comprise very few synthetic parameters. Hence the need to summarize hold also for other aspects. For example, the behaviour of a cutting-in vehicle can be realistically modelled with a couple of parameters (curation and deceleration of the manoeuver) and could not afford less important details such as, e.g., the exact

trajectory, differences in lane widths and vehicle dimensions, etc. Thus, for sub-UCs 1.2, 3.1 & 3.2, summarizing the high-level effects of sensors, communications, behaviours are necessary to maintain the logical scenario dimensions at a manageable level.

5.2 SimCenter Prescan and HEEDs

In addition, the simulation tools Simcenter Prescan and HEEDs cannot fulfill the requirements for sub-UC 1.1 with respect to the subsystem "s*imulation model validation"*. For these requirements, the tool would need to be enhanced with features or used together with other tools which (i) verify that the performed simulation is executed correctly with respect to the test scenarios, and (ii) support correlation studies with physical test results in order to assess fidelity of the simulation models.

6 CONCLUSIONS

Robust safety assurance measures play a key role in the successful deployment of CCAM systems. Within the SUNRISE project, the developed SAF will be applied for validating the safety of CCAM functionalities. As limitations of relying solely on physical testing for safety assurance of CCAM systems are existing (large effort, high cost, infeasible due to time-consuming testing), the development of a harmonized simulation framework becomes imperative to accurately validate numerous test scenarios for the defined UCs. Consequently, the subsystems of the simulation framework, their corresponding requirements and suitable software tools that can meet these requirements emerge as fundamental elements for the successful application of the SAF.

In the light of the above, this deliverable presents the findings from Task 4.3 of the SUNRISE project, which aims to identify a simulation tooling to validate the CCAM functions and systems presented in the UCs defined in WP7. The work focuses on defining the simulation tools that can realistically simulate the required subsystems (T4.1) and provide adequate interfaces for data exchange with the federated Scenario Database (SCDB) developed in Work Package 6.

It is important to note that the SUNRISE Safety Assurance Framework (SAF) is tool-agnostic. The simulation tooling specifications provided in this deliverable are not meant to prescribe a single set of tools, but rather to demonstrate a process of how suitable tools can be identified and mapped to the subsystem requirements. The SAF is designed to be flexible and adaptable, allowing project partners and future users to select the tools that best fit their specific needs and constraints.

The tool selection process applied in this deliverable is crucial for the successful development and validation of the UCs developed in WP7 of the SUNRISE project. By meticulously mapping the subsystem requirements (D4.1) and use case requirements (D4.2) to the capabilities of various simulation tools, the project consortium has ensured that the selected tools can adequately address the needs of the UCs defined in WP7.

The tool selection process involves the following key steps:

- Identifying the relevant subsystems of the SUNRISE simulation framework and their requirements (D4.1 and D4.2).
- Reviewing the capabilities of existing simulation tools and their suitability for each subsystem.
- Mapping the subsystem requirements (D4.1 section 4 and D4.2 section 5) to the features of the selected tools, ensuring comprehensive coverage.
- Addressing any gaps or limitations of the tools.

- Validating the selected tooling against the total set of use case requirements (D4.2 section 8) and generic requirements (D4.2 section 4).

The tools selected for each sub-UC were chosen based on their ability to meet the subsystem requirements (D4.1 section 4) and the use case requirements assigned to those subsystems (D4.2 section 5). For example, in sub-UC 1.1, the combination of Model.CONNECT, esmini, OBGPOM FMU 2.0, Python/RoMPaC, and Simcenter Prescan was found to adequately address the requirements related to the test case manager, environment, sensors, AD function, and vehicle dynamics subsystems. Similarly, the selection of tools for the other sub-UCs was justified based on their capabilities to fulfill the specific requirements.

While the selected tools were able to meet the majority of the subsystem and use case requirements, some gaps and limitations were identified. For instance, the lack of low-level physical models for communication and the mismatch between the simulated and physical SuTs were noted as limitations of the CarMaker tool. To bridge these gaps, the project consortium recommends exploring co-simulation approaches that can integrate more detailed communication models, as well as ensuring a robust validation process to correlate the simulation results with physical test data. To enhance the clarity of conclusions drawn above, specific examples could be incorporated, where deemed necessary, to facilitate a deeper understanding of the simulation tooling specification process for each one of the defined sub-UCs.

The contents of these conclusions are most relevant for the partners working on tasks T4.4, T7.2, and T7.3 of the SUNRISE project.

Task T4.4 (Hybrid V&V framework design) should build on the simulation tooling specifications and the identified gaps/limitations to further refine the SUNRISE SAF, ensuring a comprehensive hybrid testing approach that combines virtual and physical validation.

Tasks T7.2 (SAF PoCs design) and T7.3 (SAF PoCs development and demonstration) should utilize the selected tools and the guidelines for the tool selection process to design, develop, and demonstrate the Proofs of Concepts (PoCs) for the SUNRISE SAF. The partners in these tasks should refer to the justifications for the tool selections and the identified gaps, to ensure that their PoCs address the necessary requirements and limitations.

7 REFERENCES

1. Anders Thorsén, et al. (2023). SUNRISE D4.1 “Report on relevant subsystems to validate CCAM systems.

2. P. Weissensteiner, G. Stettinger, S. Khastgir and D. Watzenig (2023). Operational Design Domain-Driven Coverage for the Safety Argumentation of Automated Vehicles. IEEE Access, vol. 11, pp. 12263-12284, doi: 10.1109/ACCESS.2023.3242127.

3. Raul Ferreira, Jose Torres, et al. (2024). SUNRISE D4.2 “Report on mapping of use case requirements to subsystems”.

4. Ilias Panagiotopoulos, et al. (2023). SUNRISE D7.1 “CCAM use cases validation requirements”.

5. https://v4safetyproject.eu/, retrieved [05 30, 2024].

6. AVL Model.CONNECT user manual, Release 2023a.

7. https://carla.readthedocs.io/en/0.9.15/core_map/, retrieved [05 30, 2024].

8. https://carla.readthedocs.io/en/0.9.15/adv_opendrive/, retrieved [05 30, 2024].

9. https://carla.readthedocs.io/en/0.9.15/ref_sensors/, retrieved [05 30, 2024].

10. https://carla.readthedocs.io/en/0.9.15/python_api/, retrieved [05 30, 2024].

11. Simcenter HEEDS | Siemens Software retrieved [05 30, 2024].

12. Establish confidence in autonomous vehicle systems with patented Critical Scenario Creation framework - Simcenter (siemens.com) , retrieved [05 30, 2024].

13.https://www.ipg-automotive.com/fileadmin/data/know-how/media_library/pdf/1209_ATZ_OPEL_Idiada_IPG_Automotive_Simulation_based_ESC_Homologation_EN.pdf, retrieved [06 08, 2024].

14.https://www.ipg-automotive.com/fileadmin/data/events/apply_and_innovate/tech_weeks/presentations/IPG_Automotive_TECH_WEEKS_TuevSued.pdf, retrieved [06 08, 2024].

15. CarMaker 11 programmer’s guide pdf, chapt. 23.

16. https://esmini.github.io/, retrieved [04 22, 2024].

17. https://eclipse.dev/sumo/, retrieved [04 22, 2024].

18.https://github.com/openMSL/sl-1-3-object-based-generic-perception-object-model, retrieved [05 21, 2024].

19. https://github.com/carla-simulator/scenario_runner, retrieved [05 30, 2024].

20. Simcenter Prescan Software simulation platform | Siemens Software, retrieved [05 30, 2024].

21. https://www.blender.org/, retrieved [05 30, 2024].

22. CarMaker 11 Users' guide pdf, chapter 5.

23. https://leaderboard.carla.org/, retrieved [05 30, 2024].

24.https://github.com/carla-simulator/scenario_runner/blob/master/Docs/openscenario_support.md, retrieved [05 30, 2024].

25. https://carla.readthedocs.io/en/latest/ecosys_simready/, retrieved [05 30, 2024].

26. https://carla.readthedocs.io/en/0.9.15/adv_agents/, retrieved [05 30, 2024].

27. https://github.com/RISE-Dependable-Transport-Systems/WayWiseR, retrieved [06 01, 2024].

28. https://github.com/RISE-Dependable-Transport-Systems/WayWise, retrieved [06 01, 2024].

29. Mauro Da Lio, Antonello Cherubini, Gastone Pietro Rosati Papini, Alice Plebe (2023). Complex self-driving behaviors emerging from affordance competition in layered control architectures. Cognitive Systems Research, vol. 79, pp. 4-14, https://doi.org/10.1016/j.cogsys.2022.12.007.

30. https://carla.readthedocs.io/en/0.9.15/tuto_G_chrono/, retrieved [05 30, 2024]

31. https://carla.readthedocs.io/en/0.9.15/catalogue/, retrieved [05 30, 2024]

32. https://carla.readthedocs.io/en/0.9.15/adv_sumo/, retrieved [05 30, 2024]

33. https://carla.readthedocs.io/en/0.9.15/adv_ptv/, retrieved [05 30, 2024]

34.https://carla.readthedocs.io/en/0.9.15/tuto_M_generate_pedestrian_navigation/, retrieved [05 30, 2024].

35.https://www.ipg-automotive.com/fileadmin/data/applications/adas/references/MAGNA_Radar_Sensor_Simulation.pdf, retrieved [06 08, 2024].

36. Jose Gomez, Elie F. Kfoury, Jorge Crichigno, Gautam Srivastava (2023). A survey on network simulators, emulators, and testbeds used for research and education. Computer Networks, vol. 237, 110054, https://doi.org/10.1016/j.comnet.2023.110054.

37. https://www.ipg-automotive.com/fileadmin/data/applications/vehicle_dynamics/references/BOSCH_Simulation-Aided_Homologation_of_ESP_Systems.pdf, retrieved [06 08, 2024].

38.https://www.ipg-automotive.com/fileadmin/data/applications/vehicle_dynamics/references/OPEL_CAE_Based_Homologation_of_ESC_Systems.pdf, retrieved [06 08, 2024].

39.https://www.ipg-automotive.com/fileadmin/data/applications/vehicle_dynamics/references/PSA_Homologation_of_ESC_Using_Simulation.pdf, retrieved [06 08, 2024].

40. A. Cherubini, G.P. Rosati Papini, A, Plebe, A. Giugliano, M. Muro, M. Da Lio, (2024). A Subsumption Scheme for Emergent Collaboration of Self-Driving Vehicles in Intersections. Presented at IFAC Symposium on Control of Transportation Systems (17th CTS), Aya Napa, Cyprus.

41. S. S. James, S. R. Anderson and M. D. Lio (2020). Longitudinal Vehicle Dynamics: A Comparison of Physical and Data-Driven Models Under Large-Scale Real-World Driving Conditions. IEEE Access, vol. 8, pp. 73714-73729, 2020, doi: 10.1109/ACCESS.2020.298859.

42. https://carla.readthedocs.io/en/0.9.15/python_api/#carlawalkeraicontroller, retrieved [05 30, 2024].

43. https://carla.readthedocs.io/en/0.9.15/python_api/#carlawalkercontrol, retrieved [05 30, 2024].

44. Duy Son, Tong & Bhave, Ajinkya & Van der Auweraer, Herman. (2019). Simulation-Based Testing Framework for Autonomous Driving Development. 10.1109/ICMECH.2019.8722847.

45. https://github.com/openMSL/sl-1-3-object-based-generic-perception-object-model, retrieved [06 17, 2024].

46. K. Kreutz and J. Eggert (2021). Analysis of the generalized intelligent driver model (gidm) for merging situations. Presented at 2021 IEEE Intelligent Vehicles Symposium (IV21), pp. 34-41, Nagoya, Japan.

47. P. Bosetti, M. Da Lio, e A. Saroldi (2014). On the human control of vehicles: an experimental study of acceleration. European Transport Research Review, vol. 6, pp. 157–170, doi: 10.1007/s12544-013-0120-2.

48. https://www.asam.net/standards/detail/openodd/, retrieved [06 17, 2024].

49. ISO (2023). ISO 34503:2023 “Road Vehicles – Test scenarios for automated driving systems – Specification for operational design domain”.

50. P. Irvine, X. Zhang, S. Khastgir, E. Schwalb and P. Jennings (2021). A Two-Level Abstraction ODD Definition Language: Part I. Presented at 2021 IEEE International Conference on Systems, Man, and Cybernetics, Melbourne, Australia, pp. 2614-2621, doi: 10.1109/SMC52423.2021.9658751.

51. H. Nakamura et al. (2022). Defining Reasonably Foreseeable Parameter Ranges Using Real-World Traffic Data for Scenario-Based Safety Assessment of Automated Vehicles. IEEE Access, vol. 10, pp. 37743-37760, doi: 10.1109/ACCESS.2022.3162601.

52. E. de Gelder et al. (2022). Scenario Parameter Generation Method and Scenario Representativeness Metric for Scenario-Based Assessment of Automated Vehicles. IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 10, pp. 18794-18807, doi: 10.1109/TITS.2022.3154774.

53. Thomas Laurent, Stefan Klikovits, Paolo Arcaini, Fuyuki Ishikawa, and Anthony Ventresque (2023). Parameter Coverage for Testing of Autonomous Driving Systems under Uncertainty. ACM Trans. Softw. Eng. Methodol. 32, 3, Article 58, 31 pages. https://doi.org/10.1145/3550270.

54. F. Hauer, T. Schmidt, B. Holzmüller and A. Pretschner (2019). Did We Test All Scenarios for Automated and Autonomous Driving Systems?. Presented at 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, pp. 2950-2955, doi: 10.1109/ITSC.2019.8917326.

55. H. Hungar (2020). A Concept of Scenario Space Exploration with Criticality Coverage Guarantees: Extended Abstract. Presented at 9th International Symposium on Leveraging Applications of Formal Methods (ISoLA 2020), Rhodes, Greece, pp. 293-306, doi: 10.1007/978-3-030-61467-6_19.

56. C. Amersbach and H. Winner (2019). Defining Required and Feasible Test Coverage for Scenario-Based Validation of Highly Automated Vehicles," Presented at 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, pp. 425-430, doi: 10.1109/ITSC.2019.8917534.

57. M. Klischat et al. (2020). Scenario Factory: Creating Safety-Critical Traffic Scenarios for Automated Vehicles. Presented at 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), Rhodes, Greece, pp. 1-7, doi: 10.1109/ITSC45102.2020.9294629.

58. E. Rocklage et al. (2017), Automated scenario generation for regression testing of autonomous vehicles. Presented at 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, pp. 476-483, doi: 10.1109/ITSC.2017.8317919.

59. C. Medrano-Berumen and M.I. Akbas (2019). Abstract Simulation Scenario Generation for Autonomous Vehicle Verification. Presented at 2019 SoutheastCon, Huntsville, USA, pp. 1-6, doi: 10.1109/SoutheastCon42311.2019.9020575.

ANNEX 1: CARLA – OPENSCENARIO 2.0 SUPPORT

Support status

OpenScenario 2.0 is not fully supported by CARLA, although it is supported partly, and it is an ongoing process. In this section the current support of OpenScenario 2.0 by CARLA is specified at the moment that the file was delivered. It supports the OpenScenario 2.0 syntax rules, but it is missing the autonomous driving model. For example, speed () is supported to set the speed, lane () is supported to set the lane, but distance() is not supported to set the movement distance, yaw() is not supported to set the yaw angle, and lateral() is not supported to set the lateral direction within the lane.

Generic modifiers support

The movement of vehicles and pedestrians can be described with certain modifiers, including move, drive, walk, etc. The modifier is currently not mandatory as vehicles can use drive() by default, and pedestrians can use walk() by default. But currently any modifier can be used, such as vehicle.fly(), vehicle.xxx(). This is a place that needs to be regulated and prioritized. In addition, the keep() modifier is not implemented.

Movement modifiers support

Name	Meaning	Support status
speed	Set the actor's speed	Supported
position	Set the character's position along the s-coordinate of the road where the relevant actor is located	Supported
lane	Set the lane for actor movement	Supported
acceleration	Set the actor's acceleration	Supported
keep_lane	Specifies that the actor remains in the current lane	Supported
change_speed	Change the actor's speed during the current period	Supported
change_lane	Specify actor to change lane	Supported
keep_position	Make the actor maintain a fixed position during the current period	Not supported
keep_speed	Keep the actor at a fixed speed during the current period	Not supported

lateral	Set the actor's position within the lane along the transverse axis using road coordinates (t-axis)	Not supported
yaw	Set the actor's heading angle (the angle between the vehicle y-axis and the lane s-axis)	Not supported
orientation	Set the actor's direction (the angle between the vehicle's x-axis and the lane's s-axis)	Not supported
along	Make the actor move along a specified route	Not supported
along_trajectory	Make the actor move along the specified trajectory	Not supported
distance	Set the distance of actor movement	Not supported
physical_movement	Set the actor's movement to physical or non-physical (non-physical: for example, the speed changes directly from 10m/s to 20m/s without taking time)	Not supported
avoid_collisions	Allow or disallow an actor to collide with another actor	Not supported

Reserved word support

Reserved words	Meaning	Support status
action	Define behavior	Supported
actor	Actor is the most used keyword to define any entity	Supported
as		Supported
bool	An expression that returns a value of type bool, which is the value of the condition.	Supported
call		Supported
cover		Supported
def		Not supported
default		Not supported
do		Not supported
elapsed	Returns true from the first time instant when the time that passed from the start of the *event context* equals or exceeds the *duration expression* specified inside the parentheses, and all later time instants. Returns false otherwise.	Supported

emit		Not supported
enum		Not supported
event		Supported
every	Returns true at each first time instant when the time that passed from the start of the *event context* equals or exceeds a multiple of the *duration expression* specified inside the parentheses. Returns false otherwise. If an optional offset named argument is included, then the first instance (and all further instances) will be offset into the future by the given duration. Otherwise, the first instance will be at the start of the *event context*.	Supported
expression		Not supported
extend		Not supported
external		Not supported
fall	Returns true when the *bool expression* specified inside the parentheses changes values from true to false, otherwise returns false.	Supported
float		Not supported
global		Supported
hard		Not supported
if		Not supported
import		Supported
inherits		Not supported
int		Not supported
is		Not supported
it		Not supported
keep	Used to constrain variables, as actor parameter binding, can be written in the scenario scope or after with	Not supported
list		Not supported
of		Not supported
on		Not supported
one_of		Supported
only		Not supported
parallel		Supported
range		Not supported
record		Not supported
remove_default		Not supported

rise	Returns true when the *bool expression* specified inside the parentheses changes values from false to true, otherwise returns false.	
scenario		Supported
serial		Not supported
AND		Supported
string		Not supported
struct		Not supported
type		Not supported
uint		Not supported
undefined		Not supported
unit		Not supported
until		Not supported
was		Not supported
wait		Supported
with		Not Supported

Data type support

Data type category	type of data	meaning	Support status
Basic data types	bool		Not supported
	int		Not supported
	float		Not supported
	string		Supported
enumerate	enum		Not supported
Physical type/unit	mass	mass unit	Supported
	length	unit of length	Supported
	time	time unit	Supported
	angle	Angle unit	Supported
	temperature	temperature unit	Not supported
	luminous_intensity		Not supported
	electrical_current		Not supported
	amount_of_substance		Not supported
composite type	struct		Not supported
	actor		Not supported
	scenario		Not supported
	list		Supported

Actor support

Name	Support status
Vehicle	Supports adding vehicles in CARLA but does not support custom vehicles imported into .fbx.
Pedestrian	Not supported
Animal	Not supported

Map support

Supports some of CARLA's own maps but does not support importing custom .xodr format maps.

Method support

Support method definition, method calling, method inheritance.

Weather

Does not support setting weather. Currently, the azimuth and altitude angles of the sun are only set in the initialization of the OSC2ScenarioConfiguration class.

ANNEX 2: ADDITIONAL REQUIREMENT TABLES

Annex 2 contains the tables listing the additional requirements defined for sub-UCs 1.3 and 4.2 along with short descriptions.

Table with additional requirements for sub-UC 1.3

Requirement ID	Short description
N-R1.3_11	Simulation framework shall be able to have a configuration interface allowing the user to configure at least the sensor position and its basic parameters.
N-R1.3_12	Simulation framework shall be able to store at least 1 hour of video data from the camera.
N-R1.3_13	Simulation framework shall be able to have a dedicated module in analysing the scenario results and generating metrics for the test coverage evaluation.
N-R1.3_14	Simulation environment must visually represent a cooperative urban environment.
N-R1.3_15	Provide a resolution of 1.7 MP (1820 x 940)
N-R1.3_16	Horizontal field of view 110°
N-R1.3_17	Vertical field of view 47°
N-R1.3_18	Video data rate must run at 2x16 fps
N-R1.3_19	Include ISP and camera control.
N-R1.3_20	Must be installed in a bracket behind the windscreen
N-R1.3_21	Camera model must deliver an output image to a middleware that will insure the transition to other subsystems.
N-R1.3_22	The perception DF of the simulation framework shall be able to: • Detect the relevant static obstacles. • Provide the position, distance, and velocity of the detections with a high confidence level. • Detect the relevant cyclists and pedestrians. • Detect oncoming vehicles. • Detect road signals, i.e. speed limit changes. • Detect relevant stopped vehicles. • Run in real time, etc.
N-R1.3_23	Include testing the ADS on the following maneuvers in the simulation: maintain speed car following, lane centring, follow driving laws, navigate roundabouts, navigate intersections, route planning, collision avoidance, emergency braking.
N-R1.3_24	Camera model must deliver an object list with 2D positions of the objects in an image plane with corresponding BoundingBox and object class.
N-R1.3_25	Camera model must deliver an object list containing road boundaries (lane detection function) with their coordinates, type of line and number of lanes.
N-R1.3_26	The set of behaviours that the CAV shall have for the validation of the ADS. They concern obstacles, speed limits, stops, other traffic agents, and giveaway speeds.
N-R1.3_27	Apply ISOstandards and EU General Safety Regulations.
N-R1.3_28	Vehicle model must be able to adapt to a specified sensor position

N-R1.3_29	Traffic agents must be able to performe a specified manoeuver on request of the scenario generator.
N-R1.3_30	Traffic agents must be able to performe a specified manoeuver on request.
N-R1.3_31	Connectivity must be able to perform Cooperative Awareness Messages (CAMs)
N-R1.3_32	Connectivity must be able to perform Collective Perception Messages (CPMs)
N-R1.3_33	A module must be able to verify that executed simulation corresponds to the request from scenario manager.

Table with additional requirements for sub-UC 4.2

Requirement ID	Short description
N-R4.2_13	Simulation framework must be able to read as input a scenario description file in OpenScenraio format and generate a corresponding scenario environment
N-R4.2_14	Simulation framework shall be able to have a configuration interface allowing the user to configure at least the sensor position and its basic parameters.
N-R4.2_15	Simulation framework shall be able to store at least 1 hour of video data from the camera.
N-R4.2_16	Simulation framework shall be able to have a dedicated module in analysing the scenario results and generating metrics for the test coverage evaluation.
N-R4.2_17	Simulation environment must visually represent a connected perception cyber-security testing environment
N-R4.2_18	Virtual testing shall include Software-in-the-Loop (SiL), Model-in-the-Loop (MiL) and CoSim methods.
N-R4.2_19	Provide a resolution of 1.7 MP (1820 x 940)
N-R4.2_20	Horizontal field of view 110°
N-R4.2_21	Vertical field of view 47°
N-R4.2_22	Video data rate must run at 2x16 fps
N-R4.2_23	Include ISP and camera control
N-R4.2_24	Must be installed in a bracket behind the windscreen
N-R4.2_25	Camera model must deliver an output image to a middleware that will insure the transition to other subsystems.
N-R4.2_26	The perception DF of the simulation framework shall be able to: • Detect the relevant static obstacles. • Provide the position, distance, and velocity of the detections with a high confidence level. • run in real time, etc.
N-R4.2_27	The set of behaviours that the truck shall have for the validation of the automated parking functionality.
N-R4.2_28	Vehicle model must be able to adapt to a specified sensor position
N-R4.2_29	Traffic agents must be able to performe a specified manoeuver on request of the scenario generator
N-R4.2_30	Traffic agents must be able to performe a specified manoeuver on request
N-R4.2_31	Connectivity must be able to perform Cooperative Awareness Messages (CAMs)
N-R4.2_32	Connectivity must be able to perform Collective Perception Messages (CPMs)
N-R4.2_33	A module must be able to verify that executed simulation corresponds to the request from scenario manager