SAFETY ASSURANCE FRAMEWORK FOR CONNECTED, AUTOMATED MOBILITY SYSTEMS

D7.2

Safety assurance framework demonstration instances design

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	WP7: Use cases and framework demonstration instances development
Deliverable number	D7.2: Safety assurance framework demonstration instances design
Deliverable responsible	Bernhard Hillbrand, Virtual Vehicle Research
Status - Version	Final – V1.0
Submission date	22/11/2024
Keywords	Safety Assurance Framework, use cases, virtual testing, XiL testing, proving ground testing

Authors/Contributors

Name	Organisation
Bernhard Hillbrand, Gerhard Benedikt Weiss	VIF
Daniel Hassler	AVL
Sarp Yetkin	AVL TR
Bryan Bourauel	BASt
Jose Miguel Torres Camara	CAF
Christian Berger	CHAL
Gabriel Villalonga	CVC
Ilias Panagiotopoulos, Anastasia Bolovinou, Elena Daskalaki	ICCS
Eloy Collado, Marta Miranda	IDI
Thaddaeus Menzel, Thomas Amler	IDI DE
Georg Stettinger	IFAG
Jobst Beckmann	ika
Francisco Navas	RESA
Martin Skoglund, Anders Thorsén, Behrooz Sangchoolie	RISE

Jeroen Uittenbogaard	TNO
Mauro Da Lio	UNITN
Patrick Irvine	UoW
Mohammed Shabbir Ali, Mohamed Cherif Rahal, Ghada Sayari	VED

Quality Control

	Name	Organisation	Date
Peer review 1	Marcos Nieto Doncel	VICOM	24/09/2024
Peer review 2	Stefan De Vries	IDI	24/09/2024
Peer review 3	Ilias Panagiotopoulos	ICCS	24/09/2024

Version history

Version	Date	Author	Summary of changes
0.1	17/01/2024	Bernhard Hillbrand	First draft of document structure
0.2	23/09/2024	All authors	Contribution to all chapters to create a first draft version
0.3	24/10/2024	All authors	Update after internal review
0.4	07/11/2024	All authors	Update after second internal review
0.5	22/11/2024	All authors	Update after third internal review
1.0	22/11/2024	Bernhard Hillbrand	Final version ready for submission

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS AND ACRONYMS

Abbreviation	Meaning
ACC	Adaptive Cruise Control
AD	Automated Driving
ADS	Automated Driving System
AEB	Autonomous Emergency Braking
ALKS	Automated Lane Keeping System
API	Application Programming Interface
ASAM	Association for Standardization of Automation and Measuring Systems
C-ACC	Connected ACC
CAM	Cooperative Awareness Message
CarFASE	CARLA-based Fault And Simulation Engine
CARLA or Carla	CAR Learning to Act
CAV	Connected and automated vehicle
CCAM	Connected, Cooperative, and Automated Mobility
CDA	Cooperative Driving Automation
COTSATO	COncretizing Test Scenarios and Associating Test Objectives
CP	Collective Perception
CPM	Cooperative Perception Messages
D-GNSS	Differential Global Navigation Satellite System
DDT	Dynamic Driving Task
DENM	Decentralized Environmental Notification Message
DF	Data Framework
EEBL	Emergency Electronic Brake Light
ESC	Electronic Stability Control

esmini	Environment Simulator Minimalistic
ETSI	European Telecommunications Standards Institute
Euro NCAP	European New Car Assessment Programme
FCW	Forward Collision Warning
FMU	Functional Mock-up Unit
FoV	Field of View
FuSa	Functional Safety (ISO26262)
GP	Gaussian Processes
GPS	Global Positioning System
HEADSTART	Harmonised European Solutions for Testing Automated Road Transport
HWP	HighWay Pilot
ID	IDentity
IMU	Inertial Measurement Unit
INET	Open-source model library for the OMNeT++ simulation environment
ISMR	In-Service Monitoring and Reporting
KPI	Key Performance Indicator
LDM	Local Dynamic Map
LiDAR	Light Detection and Ranging
LSS	Lane Support System
MAC	Medium Access Control
MQTT	Message Queuing Telemetry Transport
ms-van3t	Multi-stack Simulation Framework for Vehicular Applications Testing
NATM	New Assessment/Test Method for Automated Driving
ns-3	network simulator-3
ODD	Operational Design Domain

OEM	Original Equipment Manufacturer
OMNeT++	Objective Modular Network Testbed in C++
OSI	Open Simulation Interface
PC	Personal Computer
PDF	Probability Density Function
POG	Probabilistic Occupancy Grid
RADAR	RAdio Detection And Ranging
RoMPaC	Robust Motion Planning and Control
ROS	Robot Operating System
RPC	Remote Procedure Calls
RSU	Road Side Unit
RX	Receiver
SAE Lx	Society of Automotive Engineers Level x
SAF	Safety Assurance Framework
SCDB	SCenario DataBase
SDL	Scenario Description Language
SiL	Software-in-the-Loop
SOTIF	Safety Of The Intended Functionality (ISO21448)
SPI	Safety Performance Indicator
SUMO	Simulation of Urban MObility
SUNRISE	Safety assUraNce fRamework for connected, automated mobIlity SystEms
SUT	System Under Test
TRL	Technology Readiness Level
TTC	Time-To-Collision
TX	Transmitter
UC	Use Case

UNECE	United Nations Economic Commission for Europe
V2V	Vehicle to Vehicle
V2X	Vehicle to Everything
V&V	Verification and Validation
VEINS	VEhicles In Network Simulation
VRU	Vulnerable Road User
VUT	Vehicle Under Test
WiFi	Wireless Fidelity
WP	Work Package
xG	x-Generation
XiL	X-in-the-Loop (e.g., SiL)
XODR	ASAM OpenDRIVE (file extension)
XOSC	ASAM OpenSCENARIO (file extension)
YOLO	You Only Look Once

EXECUTIVE SUMMARY

The **S**afety ass**U**ra**N**ce f**R**amework for connected, automated mob**I**lity **S**yst**E**ms (SUNRISE) project aims to accelerate the safe deployment of Cooperative, Connected, and Automated Mobility (CCAM) technologies by developing and demonstrating a Safety Assurance Framework (SAF). The project addresses the challenges of safety assurance in CCAM systems by focusing on a mixture of physical and virtual testing, scenario databases, validation methods, and harmonisation of standards.

This deliverable builds on the use cases presented and described in SUNRISE deliverable D7.1 [1] and on the current status of the SUNRISE SAF, which is being developed in parallel. The goal of the use cases is to design concrete SAF demonstrations by leveraging the methodologies and tools developed in WP2-WP6, aiming to validate and showcase the SAF and promote understanding and acceptance of SUNRISE's safety assurance of CCAM systems. To achieve this, the **use cases cover a wide range of application areas**, systems and driving situations (urban situations, highway driving, diverse weather conditions).

The aim of WP7 is not to validate the CCAM systems of the use cases, but to **demonstrate the SUNRISE SAF** and validate its blocks from the user perspective. In this way possible errors, gaps and improvements to the underlying methods, tools and data developed in WP2-WP6 can be found and addressed.

This document focuses on two main things. The first part (chapter 2) briefly introduces the SAF and its components, as deliverable D2.3 'Final SUNRISE safety assurance framework' with a detailed description of the framework will be published at the end of the project. It also defines the **criteria** according to which the validation of the individual blocks of the SAF will be carried out. It has to be noted that it will not be possible to fully validate the SAF within SUNRISE due to the limited number of use cases. The **blocks** that will be validated within the SUNRISE project are (see Figure 3)

- Store
- Federated Database Framework
  (Interface between the blocks "Store" and "Query & Concretise)
- Query & Concretise
- Allocate
- Execute
- Coverage
- Test Evaluate
- Decide

This part also indicates which use cases will cover each of the blocks.

The second part of this deliverable (chapter 3) then refers to the 8 **use cases** and describes their design and current development status. It looks at the current setup of the use cases and

the plans for virtual, XiL and proving ground testings. It also goes into more detail about which activities are planned in the individual SAF blocks in each of the use cases.

The actual tests and the evaluation of the demonstration of the SAF will then be the content of the next deliverable D7.3.

1 INTRODUCTION

1.1 SUNRISE project

Safety assurance of Connected, Cooperative, and Automated Mobility (CCAM) systems is a crucial factor for their successful adoption in society, yet it remains a significant challenge.

CCAM systems need to demonstrate reliability in all driving scenarios, 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 would be infeasible. In consequence, a carefully designed mixture of physical and virtual testing has emerged as a promising approach, with the virtual part bearing more significant weight in this mixture for cost efficiency reasons.

Several worldwide initiatives have started to develop test and assessment methods for Automated Driving (AD) functions. These initiatives have already moved from conventional validation to a scenario-based approach and combine different test instances (physical and virtual testing) 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 quality assurance. Nevertheless, the fact that there is a lack of a common European validation framework and homogeneity regarding validation procedures to ensure safety of these complex systems, hampers the safe and large-scale deployment of CCAM solutions. 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 (SCDBs) are another issue tackled by several initiatives and projects, that generally tends to silo solutions. A clear concrete approach should be used (at least at the European level), dealing 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 must be set and be valid for 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 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 **Safety Assurance Framework** (**SAF**) for the safety validation of CCAM systems, including a broad portfolio of Use Cases (UCs) and comprehensive test and validation tools. This will be done in **SUNRISE**, which stands for **S**afety ass**U**ra**N**ce f**R**amework for connected, automated mob**I**lity **S**yst**E**ms.

The SAF is the main element to be developed in the SUNRISE project. As Figure *1* indicates, it takes a central role, fulfilling the needs of different automotive stakeholders that all have their own interests in using it.

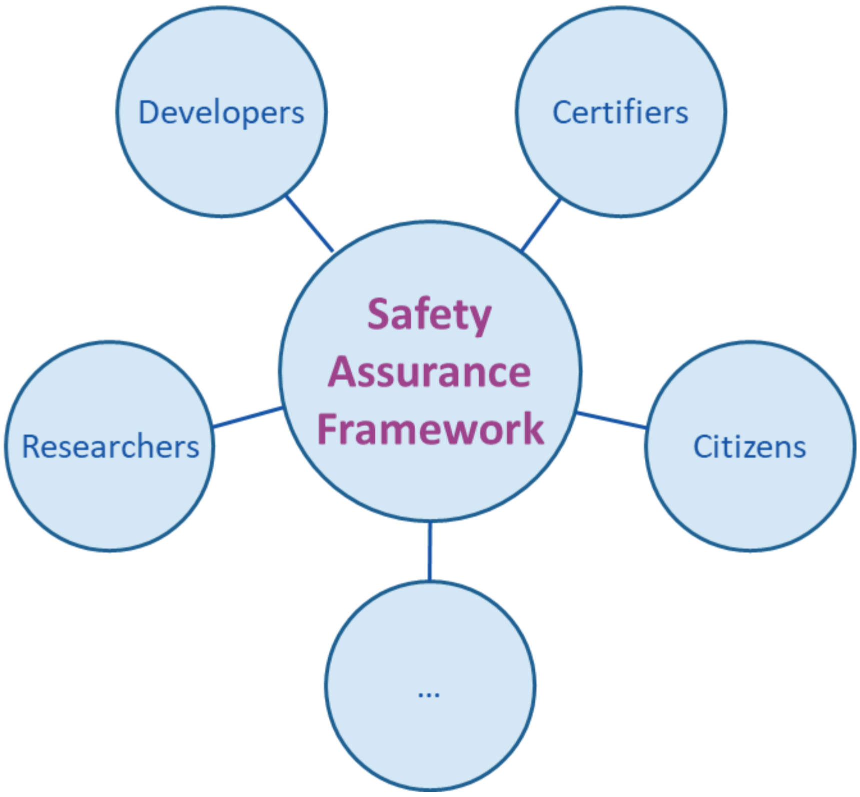
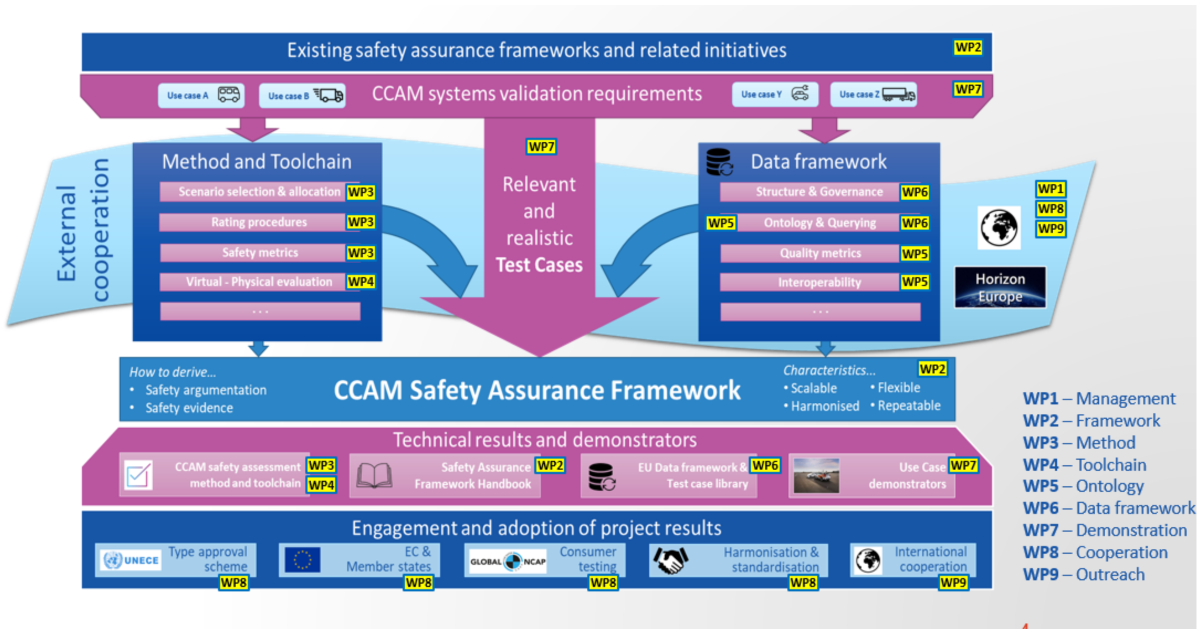


Figure 1: Safety Assurance Framework stakeholders

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 the EU's long-term goal of moving close to zero fatalities and serious injuries by 2050 (Vision Zero), and the resilience of (road) transport systems. The project 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 test cases, based on a harmonised simulation and test environment with standardised, open interfaces and quality-controlled data exchange.

Following a common approach will be crucial for present and future activities regarding the testing and validation of CCAM systems, allowing to obtain results in a standardised way, to improve analysis and comparability, hence maximising the societal impact of the introduction of CCAM systems.

Figure 2 shows the general workplan of the SUNRISE project.



Existing safety assurance frameworks and related initiatives
WP2
Use case A
Use case B
CCAM systems validation requirements
Use case Y
Use case Z
WP7
External cooperation
Method and Toolchain
Scenario selection & allocation WP3
Rating procedures WP3
Safety metrics WP3
Virtual - Physical evaluation WP4
...
WP7
Relevant and realistic Test Cases
Data framework
Structure & Governance WP6
WP5 Ontology & Querying WP6
Quality metrics WP5
Interoperability WP5
...
WP1
WP8
WP9
Horizon Europe
How to derive...
Safety argumentation
Safety evidence
CCAM Safety Assurance Framework
Characteristics...
Scalable
Flexible
Harmonised
Repeatable
WP2
Technical results and demonstrators
CCAM safety assessment method and toolchain WP3 WP4
Safety Assurance Framework Handbook WP2
EU Data framework & Test case library WP6
Use Case demonstrators WP7
Engagement and adoption of project results
UNECE Type approval scheme WP8
EC & Member states WP8
GLOBAL NCAP Consumer testing WP8
Harmonisation & standardisation WP8
International cooperation WP9
WP1 – Management
WP2 – Framework
WP3 – Method
WP4 – Toolchain
WP5 – Ontology
WP6 – Data framework
WP7 – Demonstration
WP8 – Cooperation
WP9 – Outreach

Figure 2: Wokplan of the SUNRISE Project

1.2 Purpose of the deliverable

This deliverable is part of WP7 “Use cases and framework demonstration instances development”, which focuses on the definition, design, demonstration and validation of the specified use cases. This deliverable contains the results of T7.2 and will focus on the **design** of the **demonstration instances** of the SUNRISE SAF. The content is based on the work of the other technical work packages (WP2-WP6) and how to link the outcome of this work with the demonstrators of WP7.

D7.2 should also enable T7.3 team members to fully develop and execute the UCs, targeting the validation of the SUNRISE SAF and its blocks from the user perspective, and addressing possible errors, gaps and improvements to the underlying methods, tools and data developed in WP2-WP6.

This deliverable also describes the designs of the virtual, XiL-based and proving ground testing setups and which blocks of the SAF will be covered by the individual UCs. The UCs cover a wide range of application areas, Automated Driving Function (ADFs) and driving situations (urban situations, highway driving, diverse weather conditions). This will show that the SAF can be used for a wide range of ADFs, by covering different driving situations and environments. With this, the trust and acceptance of the wider public in the safety of CCAM systems should be increased.

While the SAF will finally be used to assess the safety of a system under test (SUT), the SUNRISE use cases intend to validate the first prototype of the SAF (TRL level 6-7) instead. Therefore, it is necessary to know who will validate which SAF blocks and what are the criteria for a successful validation. It has to be noted that it will not be possible to fully validate the SAF within SUNRISE due to the limited number of use cases. The work in WP7 provides a starting point for the full validation of the SAF which is outside the scope of the SUNRISE project.

Figure *3* shows the blocks of the SUNRISE SAF that will be validated (indicated with red circles and squares).

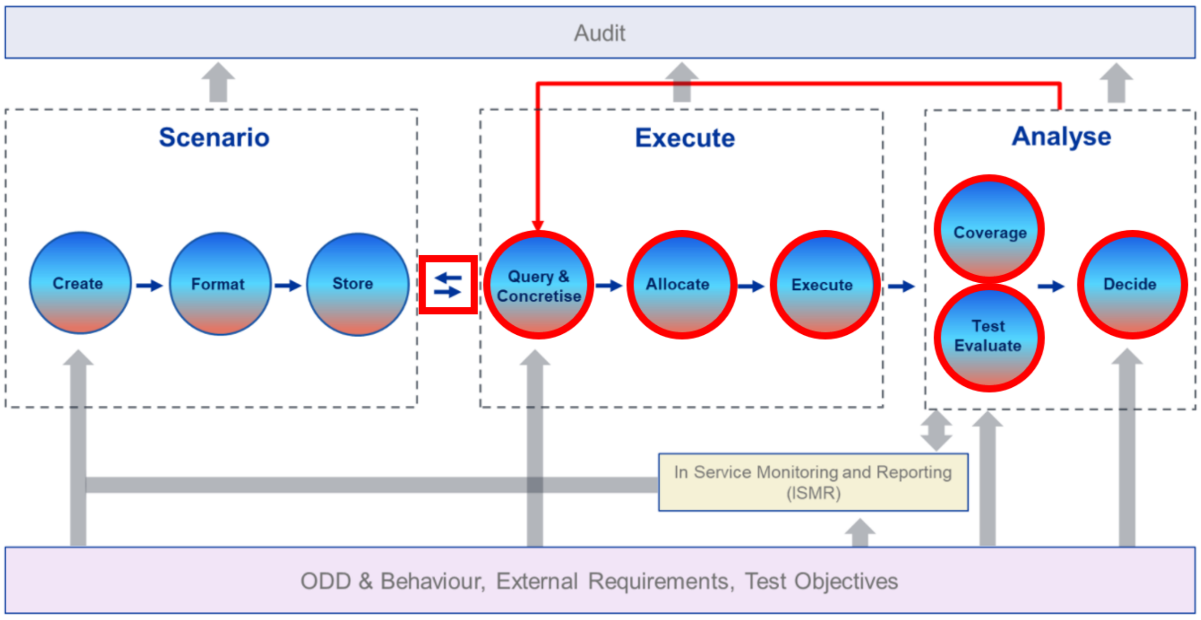


Figure 3: Draft version of the SUNRISE SAF.

1.3 Intended audience

This deliverable serves multiple stakeholders. One stakeholder is the SUNRISE project itself, as this deliverable presents the design of the demonstration instances of specified UCs that will demonstrate the functionalities and workflow of the SUNRISE SAF. Therefore, the content is of great importance for the follow up task T7.3 and the further work on the UCs.

The deliverable is also of interest to all users of the SUNRISE SAF to get more insight into the SUNRISE SAF's validation activities. The report describes which blocks of the SAF will be validated by the individual UCs and how these blocks will be validated.

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

This deliverable serves as the connection between the definition of the UCs (see D7.1 "CCAM use cases validation requirements") [1] and their demonstration (see D7.3 "Safety assurance framework demonstration instances").

The content of this deliverable is divided into the following chapters:

Chapter 2 gives an introduction into the current version of the SAF with all its blocks and a short description of them. It is not expected that the SAF will change drastically till the end of the project. However, the full description of the SAF will be published at the end of the project in D2.3 "Final SUNRISE safety assurance framework". This chapter also gives an overview of the blocks that each of the UCs will validate and the definition of when a certain block of the SAF can be considered as validated.

Chapter 3 is focused on the individual UCs. It explains the designs of the virtual, XiL-based and proving ground testing setups and the plans for validation of the individual blocks of the SAF. In addition, deviations from the UC descriptions from D7.1 [1] will be addressed. These deviations are necessary because a partner that was involved in several UCs left the consortium.

2 VALIDATION OF THE SAFETY ASSURANCE FRAMEWORK

The definition of the SUNRISE SAF started at the beginning of the SUNRISE project. Several existing methodologies, frameworks and initiatives were analysed and a draft version of the SAF was developed. This deliverable will talk about a “draft” version since this deliverable will be submitted before D2.3 “Final SUNRISE safety assurance framework” and there is a possibility that certain aspects of the SAF will change in the last year of the project.

2.1 Overview of the SAF

Building on the UNECE NATM (New Assessment/Test Method for Automated Driving), the SUNRISE SAF (see Figure 3) includes **performance assurance**, **audits**, and **in-service monitoring and reporting (ISMR)**. Performance assurance evaluates systems based on their operational design domain (ODD) and behavioural capabilities, while audits ensure that development and testing processes and tools are adequate. ISMR focuses on identifying additional edge cases and providing feedback to improve performance assurance. This document explores use cases that demonstrate the performance assurance workflow, which consists of three main components: **scenario**, **execute**, and **analyse**. The scenario component involves **creating**, **formatting**, and **storing** scenarios in databases using knowledge-based or data-driven approaches. These scenarios are structured across four levels of abstraction - functional, abstract, logical, and concrete - to address the needs of different stakeholders, with common formats like ASAM OpenX. During the execute block, scenarios are made executable through **querying and concretising** and aligned with test objectives. They are then **allocated** to suitable test environments, ranging from virtual simulations to real-world testing, where they are **executed**. The analysis block includes three crucial processes: **coverage analysis**, **test evaluation** and **decide**. Coverage analysis checks whether the scenarios adequately explore the ODD, in parallel test evaluation determines if the system meets specific safety criteria during each test. The results of these analyses are used to inform the decide block process and iteratively refine scenario parameters, leading to a comprehensive safety assessment outcome, if not reached the first time. This workflow ensures a thorough and adaptable system safety evaluation across various testing environments and for different stakeholder needs.

2.1.1 Scenario

In the “Scenario” component, three distinct steps are involved: Scenario Creation, Scenario Formatting, and Scenario Storage on a platform from which the scenarios can be accessed.

Create

Scenario creation takes place in individual SCDBs, the creation of these scenarios is not detailed as part of the SAF framework presented in SUNRISE and will not be considered in the validation efforts defined, however from a safety assurance point of view, the scenario creation process is still an integral component.

Format

Scenario formatting is a primary concern of the individual SCBDs. How a scenario is natively formatted within the SCDBs is outside of the validation efforts which accompany the SAF. However, there are requirements towards the scenario format which have been identified in D5.1 [2], should the SCDBs to be connected to the federated layer. A few general reflections on the scenario format based on the D5.1 [2] progress are that for interaction with the federated layer (which is validated in the SUNRISE SAF): scenarios need to be in a common format.

Store

The SUNRISE data framework (DF), which in this current version of the SAF is the interface between this block and the “Query & Concretise” block will query to individual SCDBs, and SCDBs will return the scenarios adhering to the identified requirements. The format of the DF and the requirements specifications for scenario queries are under development at this stage of the SUNRISE project and will be described in the deliverable D6.3.

2.1.2 Execute

The “Execute” component encompasses Querying and Concretising Test Scenarios, Allocation of scenarios to different forms of testing, and finally test Execution.

Query & Concretise

There are two main aspects of this block: the retrieval of the scenarios returned from the SUNRISE federated layer, combined with the testing objectives and other external requirements to create individual test cases. If scenarios are returned at the logical level (i.e., parameters are described in ranges), the block creates concrete scenarios with concrete parameter values, and then combine the concrete test scenarios with further test objectives. The exact form of these objectives for which validation will occur has not been identified yet.

Allocate

The development of a methodology for allocation within the Execute block of the SAF is under development currently. This block will oversee the distribution of scenarios to each execution environment (from pure simulation to real-world trials). The functional aspect of this block is currently being developed in T3.4 & T3.5.

Execute

This block includes the actual execution of the concrete test scenario within an allocated test environment.

The execute block is essentially an "external" component. It relies on established simulation systems and physical testing methods already implemented in the vehicle dynamics community. The SAF is not responsible for defining, developing, or validating these simulation systems or testing tools. At the SAF level, we can check if a particular concrete scenario has failed or produced results that seem anomalous, such as those falling outside of expected distributions.

2.1.3 Analyse

In the “Analyse” component, the results of executed test cases are evaluated through Coverage, Test Evaluate and Decide.

Coverage

Coverage analysis involves assessing the results of the test cases against coverage goals at both the test suite and logical scenario levels to determine the need for additional testing. It aims to determine the coverage of the test results; it will receive any relevant requirements from the input layer. One example of such coverage could be the coverage of the logical parameter ranges by the individual concrete parameter values. The outcome of the coverage analysis will become one of the inputs into the system decide block.

Test Evaluate

Test Evaluation consists of collecting the individual test outputs and organizing them for a global assessment (e.g., finding critical and successful logical scenario sub-regions, distributions of results and anomalies). The test evaluation block will receive input from the input layer on the external requirements, as well as the test objectives flown down from the “Execute” main block, carried by the test case. In addition, the in-service monitoring and reporting block can also further feedback to this block on any newly identified evaluation criteria. The details of this evaluation are yet to be established at this stage of the SUNRISE project.

Decide

The Decide block assesses the overall system safety based on a collection of test cases (which are representative of the ODD for the SUT evaluation). The system “Decide” block will then take the inputs from both the coverage analysis block and the “Test evaluation” block to determine the final system evaluation outcome of the test. The detailed inner workings of this block will be determined further during future discussions within the SUNRISE project.

2.2 Criteria for validation of the SAF

The main goal of the SUNRISE UCs is the validation of the SUNRISE SAF (see Figure 3). and its blocks from the user perspective, by addressing possible errors, gaps and improvements to the underlying methods, tools and data developed in WP2-WP6. To correctly check the operation of each block, criteria have been defined that must be fulfilled for a block to be called validated. Please consider that the validation of each block might depend on the validation of its prior blocks (e.g., failed validation of the “Execute” block might be caused by incomplete data from the “Query & Concretise” block or the wrong allocated test instance from “Allocate” block).

It has to be noted that the first two blocks “Create” and “Format” are not a focus of the SUNRISE project and will not be validated by the UCs.

The **SAF blocks validation criteria** are as follows:

<u>SUNRISE DF (Interface between “Store” and “Query & Concretise”)</u>

The SUNRISE DF is not a dedicated block of the SAF, but it is still important to validate it. This interface consists of two steps or two directions. The first one is the provision of the scenario query from the "Query & Concretise" block to the Data Framework (DF) (formats and definitions must match according to format and content for query in Deliverable D5.2) and the second step is the export of logical scenarios from the DF and its connected SCDBs based on the provided query and the provision of those to the “Query and Concretise” block. The exported logical scenarios must be in a format suitable to generate concrete scenarios based on the parameters, the given ranges or distributions. Also, the exported logical scenarios must match the requirements provided by the query information.

To validate the DF the following requirements must be fulfilled:

- The user has to be authorised for at least one connected scenario database (SCDB) of the SUNRISE DF and for the DF itself.

- The connected SCDBs mentioned above must contain some scenarios that shall be queried (more information on metrics for the quality of SCDBs will be in the upcoming Deliverable D5.3). The definitions in the query and the DF must align since a common understanding of scenarios or parameters is crucial for a successful query (see Deliverable D3.2 [3] chapter 4 ”Requirements for scenario concept”).

- The DF provides the expected logical scenarios with parameters and ranges or distributions
    - matching the query data and
    - in the expected format (scenario data formats are treated in section 2.6 “Data format” in Deliverable D6.1 [4])

  to be able to generate concrete scenarios.

<u>Query & Concretise:</u>

This block must provide the concrete scenarios necessary to execute tests and collect data for evaluation. The concretisation step shall follow the methodology from the upcoming Deliverable D3.4.

For each test in the batch, the Query & Concretise block must provide:

Query data: Queries should retrieve data according to the format and content outlined in D5.2 for the “Store” block, following the process defined in sections 3.2.2 and 3.2.3 of D6.1 [4].

Validation Criteria:

- Confirm that queried data aligns with the ontology structure and formats specified in D5.2.
- Ensure the returned data accurately contains all information requested in the query, with no omissions.
- Ensure that running the same query with identical parameters produces consistent results across multiple executions. This can be checked by repeatedly running sample queries to confirm reproducibility.
- Test the system's performance when handling an increasing number of queries and larger datasets. This can involve load testing or benchmarking under varying data volumes and processing loads.
- Introduce queries with unexpected or invalid input parameters (e.g., out-of-range values or malformed queries) and verify that the system handles these correctly.

Concrete scenarios: Scenarios should provide test parameters that reflect parameter distributions or defined boundaries, according to the query criteria.

Validation Criteria:

- Verify that each concrete scenario adheres to the logical scenario definition generated by the query.
- Check that each scenario's data is complete, accurate and without missing or corrupted information.
- If parameter distributions are used, confirm that sampled scenarios follow these distributions accurately.
- Conduct sensitivity analysis to determine the impact of variations in each parameter on the scenario outcome. This could involve altering parameters within a range and evaluating how significantly the scenario changes.
- For scenarios with multiple parameters that may interact or depend on each other (e.g., speed and distance), ensure that parameter values are correlated appropriately.
- Confirm that all concrete scenarios and associated data are version-controlled, with clear traceability for scenarios derived from earlier steps or iterations.

Behaviour Specification: Defines the behaviours of test participants, including the ego vehicle and other agents, specifying actions, triggers, and available communication or data (e.g., deceleration, lane changes, or turns).

Validation Criteria:

Check that all specified behaviours conform to the ontology structure defined in D5.2.

The goal of scenario concretisation is to provide a list of tests that has to be:

Efficiency: Test coverage should prioritize the regions of interest, avoiding excessive density outside these areas or uniform sampling when unnecessary.

Validation Criteria:

- Define a minimum rate of change in evaluation metrics between sampling points based on evaluation criteria and sampling methodology and validate that this rate of change is not violated.

- For iterative scenario concretisation approaches validate that the concretisation maintains or increases coverage in regions of interest without violating the minimum rate of change.

Completeness: Ensure sample point coverage is comprehensive, leaving no significant gaps in regions of interest or across the entire space if uniform sampling is used.

Validation Criteria:

Establish a maximum rate of change in evaluation metrics between sampling points to avoid under-representation and validate that this rate of change is not violated.

To validate this block, ensure that all outlined criteria are met. Additionally, assess the overall efficiency by comparing the number of generated tests against the SAF requirements. This comparison should factor in the available resources, aiming for a balance that minimizes test volume while maximizing coverage and relevance.

<u>Allocate:</u>

The allocation should be done by using the process (section 4.5 “Decision making for test case allocation to test instances”) and metrics (section 4.3 “Metrics for comparison of test case requirements to test instance capabilities” and section 4.4 “Metrics for decision making”) of D3.3 “Report on the Initial Allocation of Scenarios to Test Instances” [5]. Hence the block should focus on the “virtual simulations first” approach. This block can only be validated in combination with the following “Execute” block since the allocation process described in section 4.5 of Deliverable D3.3 [5] needs the results for scenario plausibility. The same holds for the validation process described in the upcoming Deliverable D3.5 for a potential reallocation. Additionally, both allocation processes in section 4.5 of Deliverable D3.3 [5] and upcoming Deliverable D3.5 are based on test case requirements which are derived from the test case and therefore the concrete scenario as a substantial part of it which is provided by the "Query & Concretise" block.

Validate this block of the SAF as follows:

- Estimate the metrics (sections 4.3 and 4.4) values according to the allocation process (section 4.5) proposed by the SAF in D3.3 [5].

- Evaluate whether the allocation indicated test instances of higher fidelity than necessary.

- Evaluate whether the allocation missed test instances of higher fidelity which would be needed for a potential re-allocation, especially physical tests.

- Since, D3.5 describes the process for re-allocation which includes the validation of the test case the following depends on the validation of the “Test Evaluate” block, which means if the test case can be evaluated. Evaluate the test case after execution) and check whether the test case happened as intended by making use of the intended behaviours (e.g., lane changes, turning at junctions, decelerations, communication or conflicts) from the “Query and Concretise” block. Missing data as a reason not to be able to evaluate the test case is covered by the validation of the “Query and Concretise” block. If the test case has not happened as intended or if the fidelity of the result does not match the expectations from initial allocation the process for re-allocation from D3.5 must be validated as part of the Allocate block. This means to follow the process for re-allocation from upcoming D3.5 and to repeat the upper steps.

Execute:

The test execution might happen in virtual, hybrid or real-world depending on the test instance allocated by the “Allocate” block. Some exemplary virtual simulation frameworks as well as the proposed harmonised V&V simulation framework can be found in Deliverable D4.4 [6].

To validate the “Execute” block the test case and therefore the test run must be prepared (e.g., models, toolchain, measurement equipment, vehicle, staff). After the test execution, this block of the SAF is validated by the following sanity checks:

- The test run has been prepared, executed and generated log and result data. The result data contains all the information needed for the calculation of all the metrics in the “Test Evaluate" block.

- Check whether log data reveal any potential errors that occurred during test preparation or execution (sanity check).

- Check whether the result data contains all the information needed for the calculation of the metrics in the “Test Evaluate" block and metrics for test case validation according to upcoming Deliverable D3.5. See also the last validation criteria of the “Allocate” block to validate the re-allocation process.

More detailed information on validation of virtual simulation frameworks will be found in upcoming Deliverable D4.5 and of hybrid or proving ground testing in upcoming Deliverable D4.6.

Due to human errors (e.g., bugs in simulation including models) or hardware (e.g., needed voltage not achieved), it might be the case that no log or result data is available. Therefore, ensure the correct implementation of the test case (including the scan for errors in log files if available) and the correct test execution first (e.g., review of written protocols if applicable), before drawing any conclusion on the validation result of the “Execute” block.

Coverage:

The “Coverage” block must provide information on the coverage of the sample points and regions compared to the whole scenario/ODD parameter space. The coverage grading must be related to the ODD or query criteria derived from it which indicate the parameter space. Finally, for a potential feedback loop from Analyse blocks to “Query & Concretise" block if the coverage is not sufficient, information about gaps in parameter space must be provided as well as of single sample points or clusters regarding potential regions of interest (e.g., edge cases of region of interest as will be described in upcoming Deliverable D3.4). Independent of the coverage, outlier information should be provided within the feedback loop for a potential resampling.

To validate the “Coverage” block the criteria are as follows:

- Compare the coverage of known parameter spaces (fixing most of the parameters) to the result of the “Coverage" block. Apply additional sanity checks by adding artificial gaps to parameter space for lower coverage and increasing the density of sample points for higher coverage. Metrics for coverage grading will be found in the upcoming D5.3 (see scenario coverage and completeness of data).
- Change results of single sample points for outlier detection and check whether appropriate feedback to the outliers was provided.
- Define region(s) of interest with lower density compared to other regions and check whether the right feedback was provided to achieve higher sampling density in the according regions.

Test Evaluate:

The “Test Evaluate" block has two main objectives, the first and most important one is the evaluation of the safety performance of the SUT for a single test case. Second, the block must provide information on the validity of the test run. Therefore, the following criteria need to be checked:

- Check all state-of-the-art safety related metrics (such as THW, TTC, distances, collisions or overlapping boundary box areas, accelerations) for applicability and check whether a plausible conclusion on the safety performance of the SUT for the single test case can be derived.
- Check whether the intended behaviours (e.g., lane changes, turning at junctions, decelerations, communication or conflicts) were observed and therefore the metrics to validate the test run derived from the test case requirements (see upcoming Deliverable D3.5) can be calculated.

Decide:

The “Decide” block must provide the safety evidence of the SUT based on the provided coverage information coming from the “Coverage" block and test evaluation results coming from “Test Evaluate" block. Therefore, the guidelines in the upcoming Deliverable D2.3 need to be followed and checked for applicability and feasibility. The metrics in upcoming D2.3 need to be applied and it must be checked if safety evidence of the SUT can be derived based on the test evaluation results and coverage analysis. An additional validation step but not in the scope of SUNRISE might be to check whether the conclusion on the safety performance of the SUT contradicts the outcome of other state-of-the-art safety assessment frameworks, ISO standards (such as ISO26262 or ISO21448) or regulations (UN R157 - ALKS). More precise validation criteria cannot be provided at this stage due to the early stage of the "Decide" block and the upcoming Deliverable D2.3.

2.3 Covered SAF blocks by use cases

The table below gives an overview of which of the SAF blocks will be covered by the activities of each use case. Details on the design and validation plans of each Use Case are described in chapter 3.

Table 1: Overview of the planned validation activities per use case

	UC1.1	**UC1.2**	**UC1.3**	**UC2.1**	**UC3.1**	**UC3.2**	**UC4.1**	**UC4.2**
SUNRISE DF				x	x	x		
Query & Concretise	x	x	x	x	x	x	x	x
Allocate	x	x	x	x	x	x	x	
Execute	x	x	x	x	x	x	x	x
Test Evaluate	x	x	x	x	x	x	x	x
Coverage		x	x		x	x		
Decide	x	x	x	x	x	x	x	

3 USE CASE DEMONSTRATION DESIGN

This chapter takes a look at each of the eight sub use cases (Table *2*). It gives an overview of the design of the use cases and the plans to validate the different blocks of the SUNRISE SAF.

Table 2: Overview of the SUNRISE use cases

UC Nr.	Name	Partners
UC1	**Urban AD validation**	
UC1.1	*Perception testing*	*CAF, IFAG, RSA, CVC, VIF, RESA, SISW*
UC1.2	*Connected perception testing*	*VED, UNITN, ika*
UC1.3	*Cooperative perception testing*	*ICCS, VED*
UC2	**Traffic jam AD validation**	
UC2.1	*Safety assessment & decision making*	*AVL, AVL TR, CAF, AVL HU, TNO, (UoW), BASt*
UC3	**Highway AD validation**	
UC3.1	*Map based perception & decision making*	*IDI, IDI DE, CRF, UNITN, UoW, ika*
UC3.2	*Cooperative perception & decision making & control*	*IDI, IDI DE, CRF, UNITN, UoW, ika*
UC4	**Freight vehicle automated parking validation**	
UC4.1	*Truck low-speed perception & decision making*	*RISE, CHAL*
UC4.2	*Truck low-speed connected perception cyber-security*	*ICCS, RISE*

3.1 Use case 1.1: Urban AD validation – Perception testing

In Use Case 1.1, the SUNRISE SAF is tested for perception systems in urban environments. The aim here is to test those systems in complex urban intersections/scenarios and adverse weather conditions. There are camera-based, LiDAR-based and radar-based perception systems.

There are six behaviours defined for UC1.1: Car-following, outdoor parking with other vehicles entering or leaving parking spots, obstacle avoidance due to construction work, pedestrian and bicycles, and mainly urban intersections like roundabouts and right/left turn junctions. All of them with day/night conditions and/or adverse weather conditions like fog, rain and others. For more information about this and other use cases see the SUNRISE deliverable D7.1 “CCAM use cases validation requirements” [1].

3.1.1 Setup

3.1.1.1 Virtual testing

Several partners provide their own simulation frameworks that will focus on different parts of the urban perception use case.

Infineon (IFAG) is dedicated to conducting radar-based perception tests in urban environments. To ensure a comprehensive approach, the project will draw upon the ODD definition, high-level validation requirements, KPIs, and metrics for assessment as outlined in WP7 Task 7.1. This will provide a solid foundation for evaluating the performance of ADS-equipped vehicles in complex urban scenarios. Furthermore, the testing scenarios will be informed by the Euro NCAP protocols, specifically the right turn scenario as described in the Euro NCAP test protocol for Autonomous Emergency Braking (AEB) / Lane Support System (LSS) Vulnerable Road User (VRU) systems [7]. This protocol has been developed to assess the capabilities of advanced driver-assistance systems, and its inclusion in this project will help to ensure that the testing is both rigorous and relevant. More specifically, the focus will be on three distinct scenarios: (i) Car-to-Pedestrian Farside Adult (see Figure 4), (ii) Car-to-Pedestrian Nearside Adult (see Figure 5), and (iii) Car-to-Pedestrian Turning Adult 4 (see Figure 6), each of which presents unique challenges and opportunities for validation in urban environments. By concentrating on these scenarios, the aim is to gain a deeper understanding of how ADS-equipped vehicles interact with vulnerable road users, such as pedestrians, in a variety of situations.

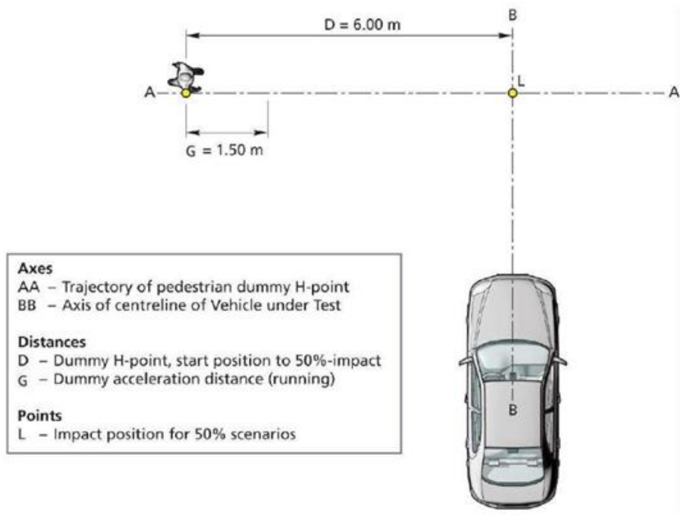

D = 6.00 m
B
A
L
A
G = 1.50 m
Axes
AA – Trajectory of pedestrian dummy H-point
BB – Axis of centreline of Vehicle under Test
Distances
D – Dummy H-point, start position to 50%-impact
G – Dummy acceleration distance (running)
Points
L – Impact position for 50% scenarios
B

Figure 4: Car-to-Pedestrian Farside Adult *[7]*

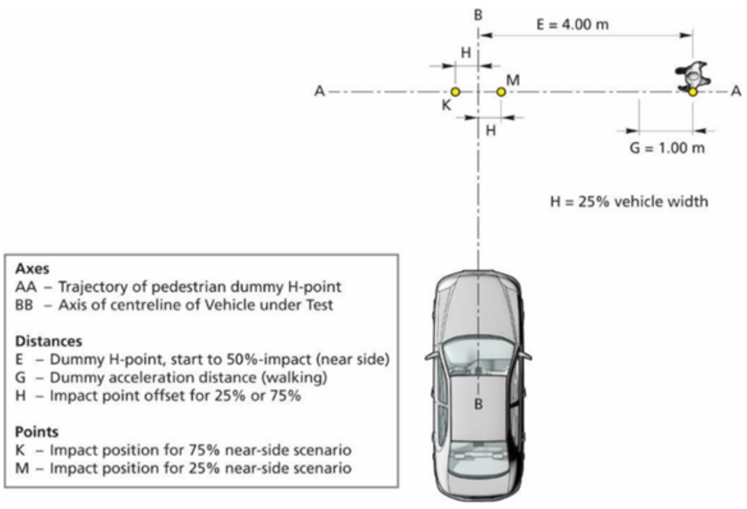

B
E = 4.00 m
H
M
A
K
H
A
G = 1.00 m
H = 25% vehicle width
Axes
AA – Trajectory of pedestrian dummy H-point
BB – Axis of centreline of Vehicle under Test
Distances
E – Dummy H-point, start to 50%-impact (near side)
G – Dummy acceleration distance (walking)
H – Impact point offset for 25% or 75%
Points
K – Impact position for 75% near-side scenario
M – Impact position for 25% near-side scenario
B

Figure 5: Car-to-Pedestrian Nearside Adult *[7]*

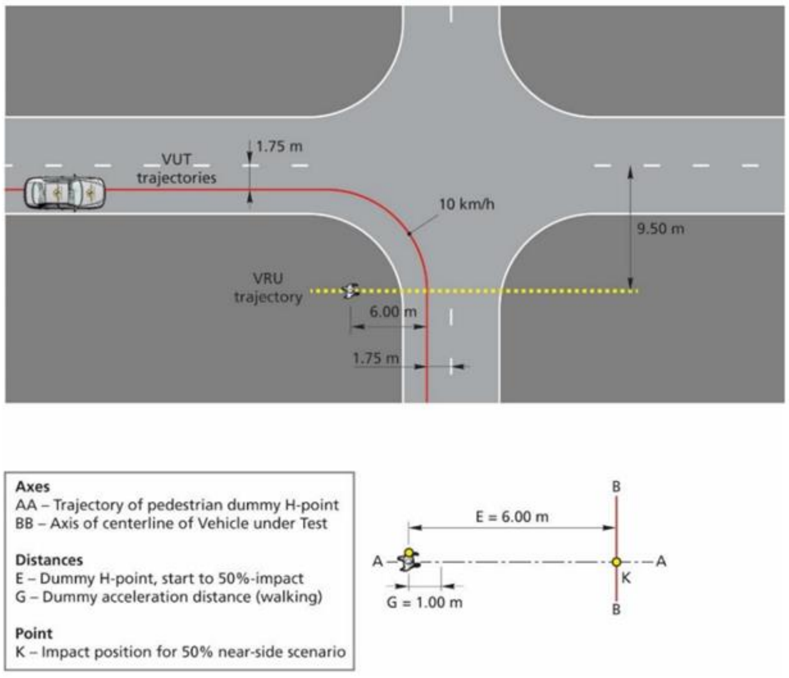

1.75 m
VUT trajectories
10 km/h
9.50 m
VRU trajectory
6.00 m
1.75 m
Axes
AA – Trajectory of pedestrian dummy H-point
BB – Axis of centerline of Vehicle under Test
Distances
E – Dummy H-point, start to 50%-impact
G – Dummy acceleration distance (walking)
Point
K – Impact position for 50% near-side scenario
B
E = 6.00 m
A
A
K
G = 1.00 m
B

Figure 6: Car-to-Pedestrian Turning Adult 4 *[7]*

The simulation framework, as specified in WP4 Task 4.4, serves as the foundation for conducting virtual tests, providing a robust and reliable platform for validating the safety of ADS-equipped vehicles. This comprehensive framework is comprised of two primary components: a high-fidelity radar sensor model, and the environment model. This environment model, for this specific realization of the framework, includes the vehicle dynamics, a rudimentary driving function, and a scenario engine. These components work in tandem to create a realistic, relevant and valid simulation environment, allowing for the testing of various scenarios and use cases in a controlled and repeatable manner and can be seen in Figure *7*. Notably, the two main subsystems of the simulation framework (the radar sensor model and the environment model) have undergone rigorous verification and validation during WP4 Task 4.5. This involved a series of focused activities, including the validation of the radar sensor model behaviour accurately reflected real-world performance. Additionally, the validation of environment-radar sensor interactions was conducted to guarantee that the simulation framework correctly captured the complex relationships between the radar sensor and the surrounding environment. Furthermore, the model correlation of raw data was performed to verify that the simulation framework could accurately process and interpret raw sensor data. Through these efforts, the simulation framework has been thoroughly tested and refined, ensuring that it provides an accurate and trustworthy basis for virtual testing and evaluation, and ultimately supporting the development of more advanced and reliable ADS.

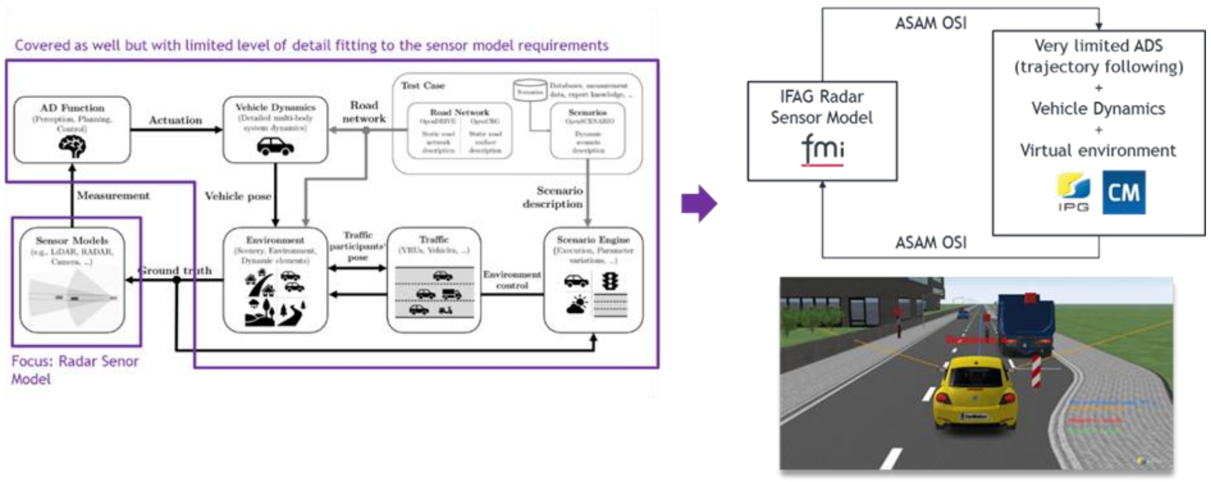

Covered as well but with limited level of detail fitting to the sensor model requirements
AD Function (Perception, Planning, Control)
Actuation
Vehicle Dynamics (Detailed multi-body system dynamics)
Road network
Test Case
Road Network
Scenarios
Measurement
Vehicle pose
Scenario description
Sensor Models (e.g., LiDAR, RADAR, Camera, ...)
Ground truth
Environment (Scenery, Environment, Dynamic elements)
Traffic participants' pose
Traffic (VRUs, Vehicles, ...)
Environment control
Scenario Engine (Execution, Parameter variations, ...)
Focus: Radar Senor Model
ASAM OSI
IFAG Radar Sensor Model
fmi
Very limited ADS (trajectory following)
+
Vehicle Dynamics
+
Virtual environment
IPG
CM
ASAM OSI

Figure 7: IFAG simulation framework

The demonstration instance is specifically designed to highlight the impact of several critical physical sensor effects and phenomena on the performance of ADS-equipped vehicles and can be seen in Figure *7*. As outlined in detail in WP4 Task 4.5, these effects include phase noise, which can introduce random fluctuations in the radar signal, compromising its accuracy and reliability. Another key phenomenon is mixer non-linearity, particularly third-order intermodulation, which can lead to the generation of spurious signals that can interfere with the radar's ability to accurately detect and track targets. Furthermore, the demonstration instance will also focus on the effects of interference from other RADAR sensors, which can be a significant issue in complex urban environments where multiple radar systems may be operating simultaneously. Finally, the instance will examine the impact of phase drift of transmitter (TX) and receiver (RX) channels, which can cause the radar signal to become desynchronized, leading to errors in target detection and tracking. With the inclusion of these physical sensor effects and phenomena, this demonstration instance - which provides a deeper understanding of the challenges and limitations of radar-based perception systems in ADS-equipped vehicles operating in complex urban environments - is able to act as relevant test method to allocate the scenarios to.

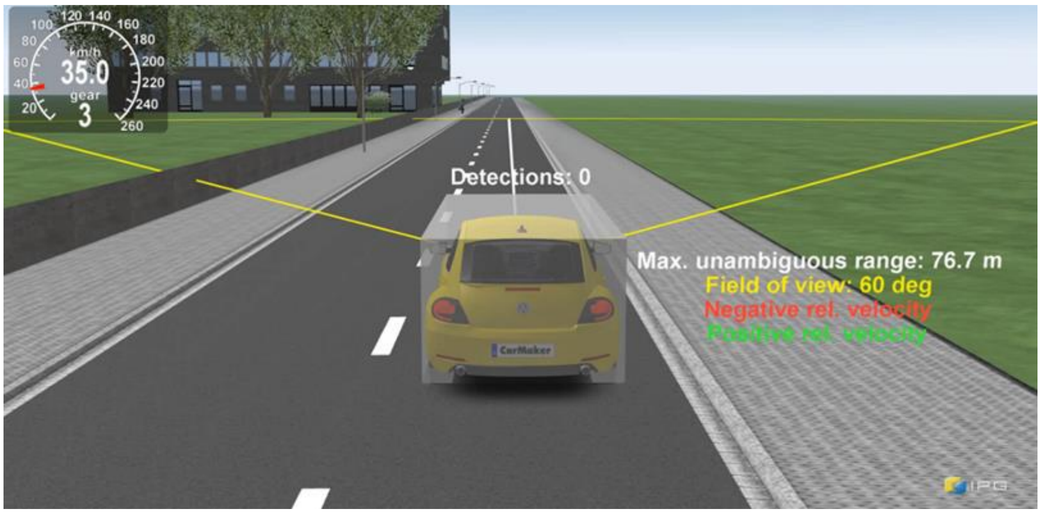


Figure 8: Exemplarily demonstration instance (Euro NCAP scenario: Car-to-Pedestrian Farside Adult).

Virtual Vehicle (VIF) will focus on left turn and right turn junction scenarios (see Figure 9 and Figure 10).

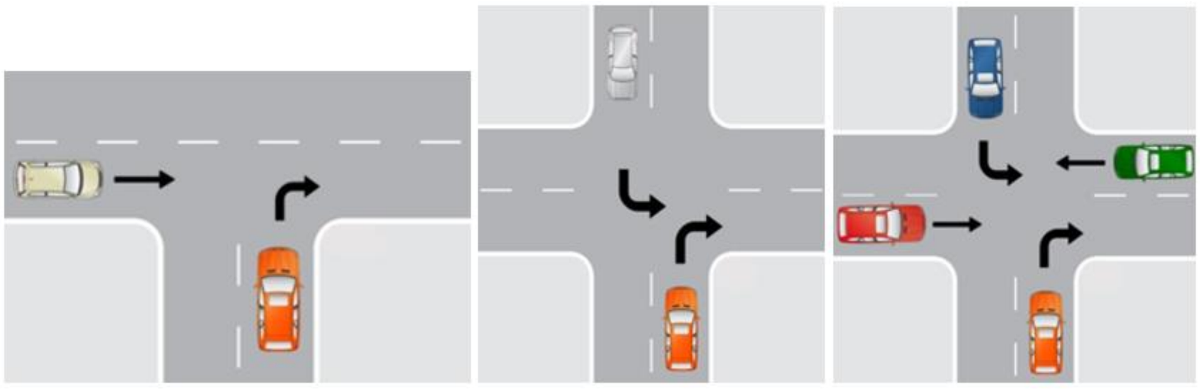


Figure 9: Right turn junction scenarios

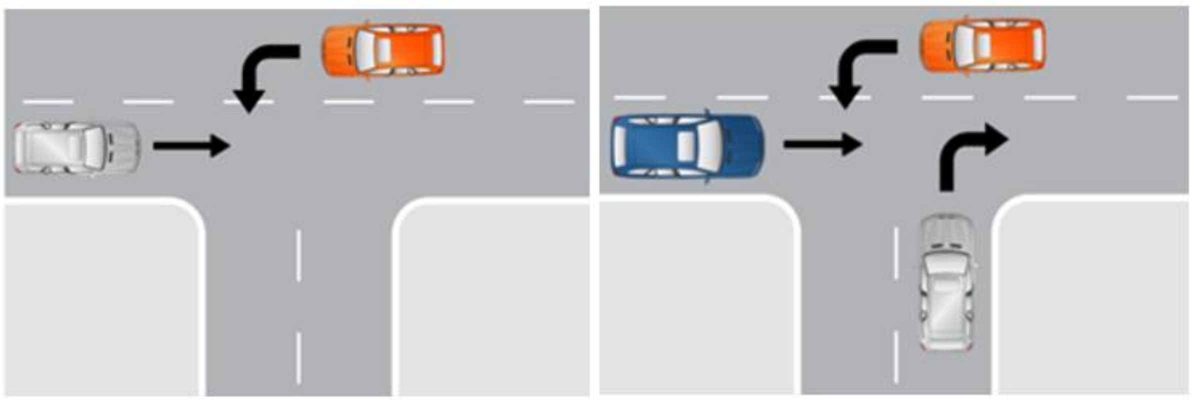


Figure 10: Left turn junction scenarios

All those scenarios will be simulated using a medium fidelity lidar model for the perception. The input and output of the sensor model is an object list defined by the ASAM OSI standard

[8]. The model has been developed by VIF. The structure of the model can be seen in Figure 11.

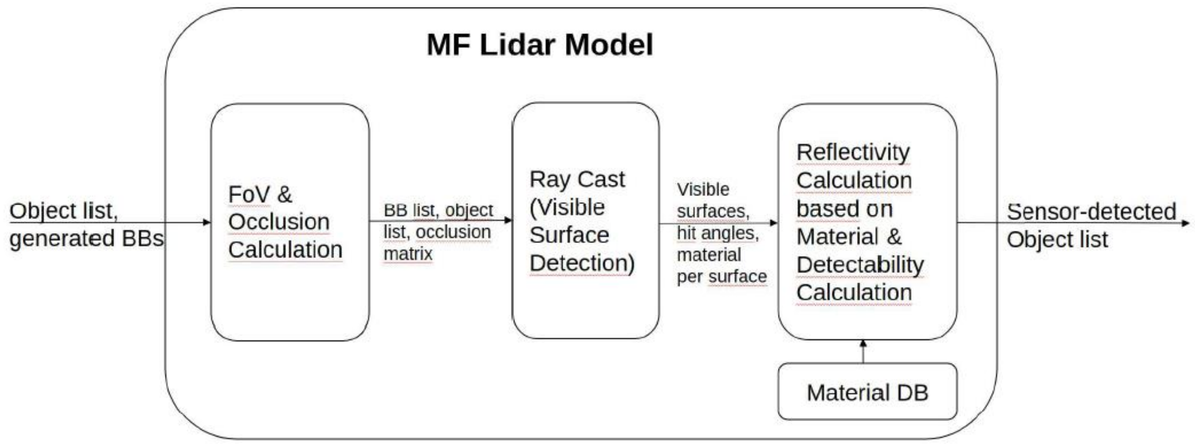

MF Lidar Model
Object list, generated BBs
FoV & Occlusion Calculation
BB list, object list, occlusion matrix
Ray Cast (Visible Surface Detection)
Visible surfaces, hit angles, material per surface
Reflectivity Calculation based on Material & Detectability Calculation
Material DB
Sensor-detected Object list

Figure 11: Structure of the sensor model used in the VIF simulation framework

The first part of the structure is a FoV (Field of View) and Occlusion check. That part is a basic check if an object is within the parameters of the sensor concerning the range and angle. The occlusion filter determines if an object is occluded by any other object.

The 2nd part is the ray casting model which uses a well-known principle to determine whether a ray hits a surface and in which angle. Figure 12 shows an example of that.

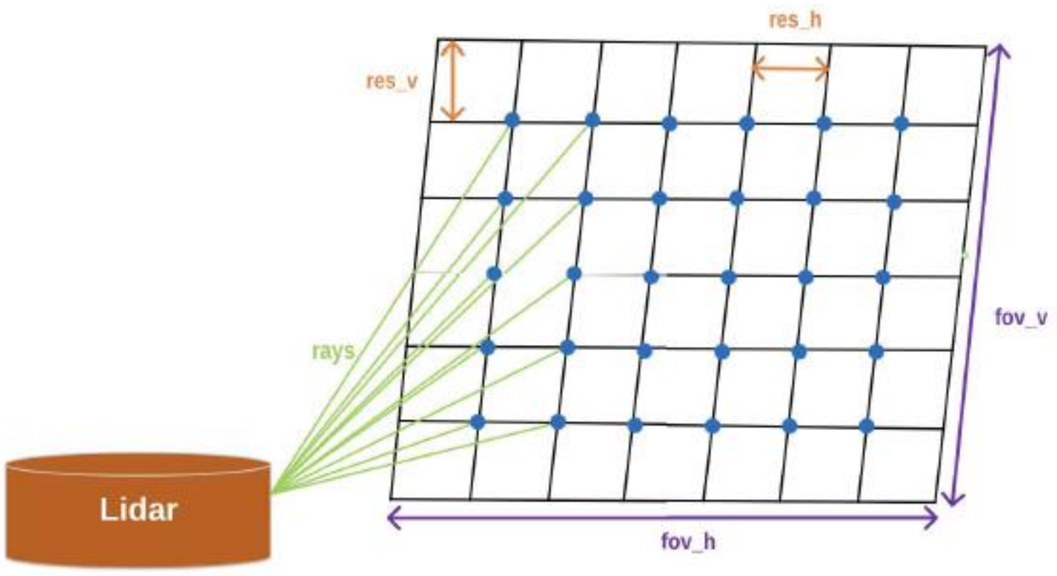


Figure 12: Ray casting model used in the VIF sensor model.

The last part is the reflectance and detection probability calculation. Lidar can measure the strength of a returning signal and use this data to determine the spectral reflectance of the surface it has just encountered. Reflectance is an angle-dependent physical property of the surface material. Whether Lidar can detect a hit largely depends on the distance and angle of the object, as highly reflective surfaces are detectable from much farther away than low-reflective ones. This model utilizes a material database to gather surface reflectance information based on the angle and determine if the surface would be detected by Lidar, depending on its distance (refer to the Concept of detection calculation).

Based on the harmonised V&V simulation framework from SUNRISE Deliverable D4.4 [6], ViF has implemented the framework in Model.CONNECT to execute the scenario simulations. Figure 13 shows the simulation topology and Figure 14 the parameters for the sensor model which is compiled as FMU 2.0.

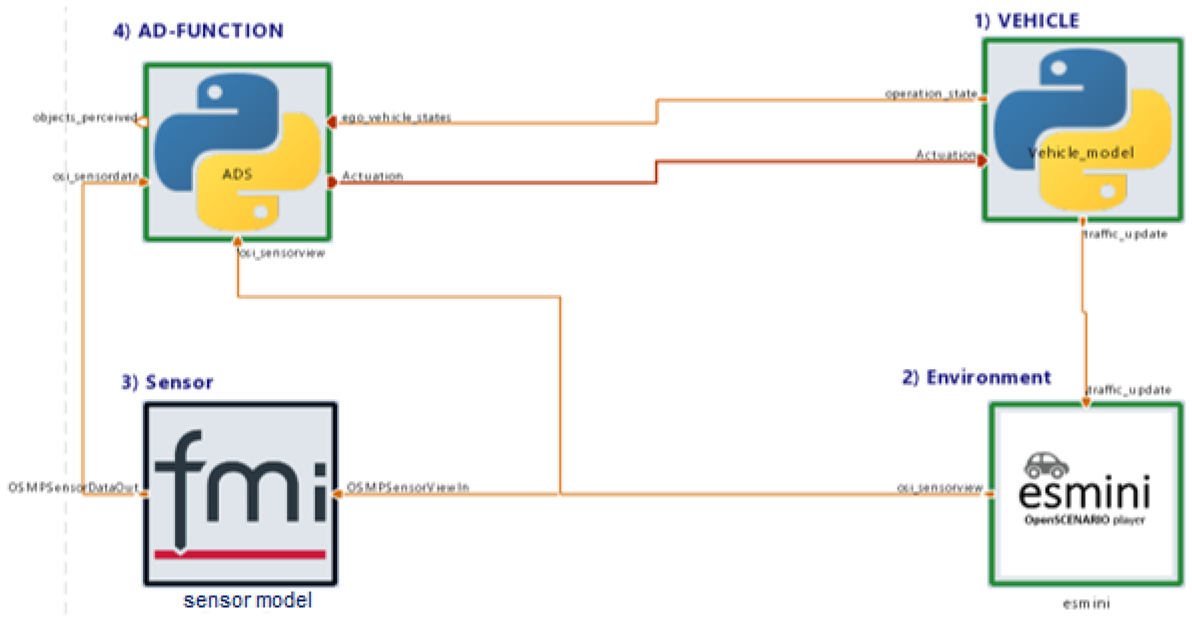
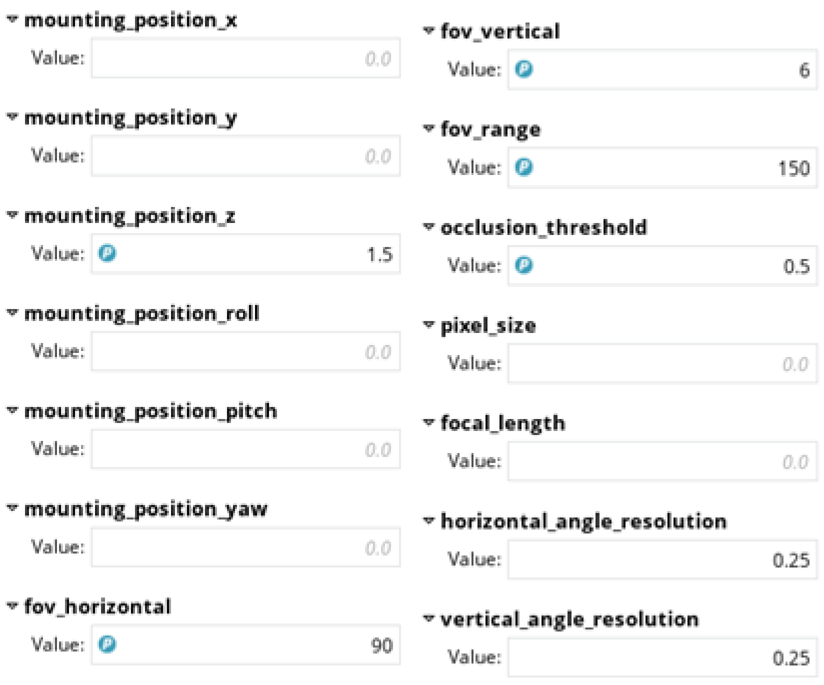


Figure 13: VIF simulation framework implemented in Model.CONNECT

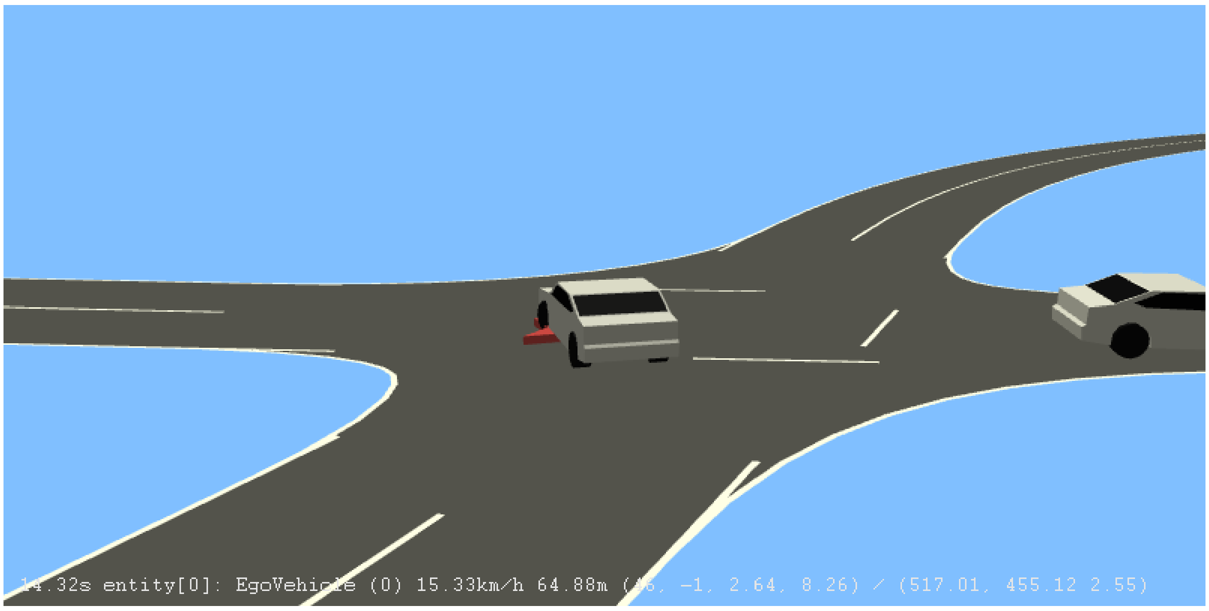


mounting_position_x	0.0
mounting_position_y	0.0
mounting_position_z	1.5
mounting_position_roll	0.0
mounting_position_pitch	0.0
mounting_position_yaw	0.0
fov_horizontal	90
fov_vertical	6
fov_range	150
occlusion_threshold	0.5
pixel_size	0.0
focal_length	0.0
horizontal_angle_resolution	0.25
vertical_angle_resolution	0.25

Figure 14: Parametrization Lidar modell

ViF uses an internally developed python module called RoMAPaC for the AD-Function and esmini as environment and traffic simulation. In order to control the ego vehicle by RoMPaC and not by esmini an external controller has to be assigned to the ego vehicle in the OpenScenario file. Thus, only the initial position of the ego vehicle is defined but not the desired route. For that reason, the OpenDrive file of the scenario has to be loaded in an OpenDrive viewer. The user has to define the desired route for the Ego Vehicle by the road and lane ID's. RoMPaC calculates with that information the line for the Ego Vehicle

Figure 15 shows the result of the simulation. The vehicle in the middle is controlled by RomPAC following the pre-defined Road IDs, the other car on the right is controlled by esmini. A simple route from left to right has been defined in the OpenScenario file for that example.



```
14.32s entity[0]: EgoVehicle (0) 15.33km/h 64.88m (46, -1, 2.64, 8.26) / (517.01, 455.12 2.55)
```

Figure 15: Example simulation of the VIF simulation framework with the graphical output of esmini.

CVC will be responsible for conducting virtual testing of camera systems in Use Case 1.1 (UC1.1). These tests will be carried out using the CARLA driving simulator and a collection of photorealistic images, as detailed in SUNRISE deliverable D7.1 [1]. Both tools will exclusively focus on the use of cameras.

The scenarios used in CARLA will be provided in OpenScenario 2.0 format, encompassing various situations such as roundabouts, object avoidance, and turns. Scenarios will be loaded either from public CARLA maps or via OpenDrive files to generate the road infrastructure. It is important to note that the environmental quality of scenarios created using OpenDrive files may be lower than that of default CARLA maps. These files will serve as inputs to define the agent's behaviour and the scenario environment for ScenarioRunner, which will be employed to execute the driving agent in each test scenario.

The driving agent will have the option to utilize either privileged information or perception models to perceive its environment. When privileged information is used, perception errors derived from the photorealistic data will be introduced to the agent, allowing for the evaluation of the driving performance based on the perception errors. In contrast, when using perception models within CARLA, the photorealistic data will not be applied.

SUNRISE deliverable D4.3 [9] specifies the subsystems and which CARLA tool is responsible for each subsystem involved in running the testing scenario.

Upon completion of the evaluation, a summary will be provided, outlining the success of the test and any errors encountered during the driving process. These errors may include collisions with objects or violations of traffic laws. The driving performance scores will be used to assess whether perception errors contribute to driving issues. Additionally, perception metrics will be provided for the perception models, covering aspects such as object detection accuracy and semantic errors.

CAF uses a SUNRISE-inspired simulation system that can generate specific urban scenarios taking as input logical ones, which specify the general features of what shall be simulated, such as how many road users will take part in the scenario, which manoeuvres are to be done by them, etc. Then, we can choose between a distribution-based sampling or a semi-random Latin Hypercube (whose output is constrained to avoid unrealistic scenarios, such as vehicles that overlap in the initial conditions). Details of CAF’s choices for sampling methods were shared with the rest of SUNRISE partners, specifically within the context of task T3.3. After running each of the generated scenarios, a criticality metric is calculated, based on speeds, distances between vehicles, etc. This helps us detect the most dangerous situations, which may require more analysis or even additional sampling around their initial conditions to validate the dangerous situations as thoroughly as possible or to detect issues within either the system under test or the simulation itself. CAF’s simulation framework is built upon CARLA, so this can be done in all the scenarios that appear in CARLA’s maps, which include a series of urban environments. CAF will not validate any part of the SAF on its own but rather support other partners to help them do so.

3.1.1.2 Proving ground testing

RESA is providing an autonomous driving prototype equipped with a LiDAR-based perception system. Proving ground mimicking an urban area with roundabouts, parking, and junctions is in Renault’s facilities in Valladolid (Spain). The vehicle’s equipment and area of operation details are in SUNRISE deliverable D7.1 [1].

This proving ground will be only used by the prototype and other agents needed to test the different scenarios (one extra pedestrian or vehicle). A safety driver will be on-board for safety reasons, taking over control of the vehicle if something fails. While the goal of the demonstration is the validation of the SAF blocks, the aim of the test is to test the perception system, but the vehicle will be equipped with a full autonomous driving pipeline, providing AD Level 4 capabilities.

The autonomous driving prototype is equipped with GPS and IMU, as well as other agents are equipped with GPS, to obtain precise positions of ego and other vehicles, evaluating the performance of the corresponding perception system when detecting other vehicles and pedestrians.

In addition to RESA physical testing will be performed by ika in this Use Case. For this physical testing ika will provide an autonomous vehicle using ika's autonomous driving software stack. The testing itself will be performed at the Aldenhoven Testing Center. Here, an urban setting is available to test the use case at an intersection with traffic lights as specified in SUNRISE deliverable D7.1 [1].

3.1.2 Covered aspects of the SAF

The table below shows the contributions to the SAF block validation per partner for UC1.1.

Table 3: Contributions to the SAF validation per Partner in UC1.1

Partner/ Block	CVC	IFAG	RESA/RSA	SISW	VIF
SUNRISE DF					
Query & Concretise	x	x	x	x	x
Allocate	x	x	x	x	x
Execute	x	x	x	x	x
Test Evaluate	x		x		
Coverage					
Decide		x	x	x	

Query & Concretise

CVC: CVC will create the scenarios defined in D7.1 [1] in CARLA with different parametrization. The method described in D3.4 will be used to create and sample the scenarios.

IFAG: IFAG will use a knowledge-based approach to creating test cases (Euro NCAP scenarios, which are based on expert knowledge and represent an important aspect to consider for supporting ADS validation). This approach enables IFAG to demonstrate the capability of handling knowledge-driven scenarios as well, addressing the needs of many vehicle safety bodies, especially for urban-based environments. This approach actively supports the validation of this SAF block.

RESA/RSA: RESA will use a sampling method described in D3.4 to create test cases related to the scenarios defined in D7.1 [1]. RESA will check that the resulting test parameters, behaviours of the other participants and quantities to be measured are aligned with the provided ODD, external requirements and test objectives. RESA will also check that there is not any missing critical scenario.

SISW: Siemens will use two of the test cases that are defined by VIF and utilize same sampling method defined in D3.4 for parametrization.

VIF: VIF will use a sampling method described in D3.4 to create test cases that can be allocated in the following block.

Allocate

CVC: CVC only will use virtual testing using CARLA. The process described in D3.3 [5] for the initial allocation will be used.

IFAG: IFAG will contribute to the "Allocate" block of the SAF by using the initial allocation process (D3.3 [5]). Since IFAG uses only virtual testing in this use case the allocation process will not be used to distribute test cases between different test instances but to verify that a test case can be allocated to the available test instance. Furthermore, since IFAG uses knowledge-driven scenarios (Euro NCAP, see entry above), this enables to demonstrate that the allocation block of the SAF is able to handle these types of scenarios as well.

RESA/RSA: RESA will contribute to the "Allocate" block of the SAF by using the initial allocation process (D3.3 [5]). As RESA only uses proving-ground testing in this use case the allocation process will not be used to distribute test cases between different test instances but to verify that a test case can be allocated to the available test instances.

SISW: Siemens plans to only use virtual testing in this use case with Simcenter Prescan. Therefore, the allocation process will be used not to distribute test cases across different tests but to confirm that each test can be assigned to an available test instance.

VIF: ViF will use the initial allocation process described in D3.3 [5]. Since VIF uses only virtual testing in this use case the allocation process will not be used to distribute test cases between different test instances but to verify that a test case can be allocated to the available test instance.

Execute

CVC: CVC will use the framework specified in D4.4 [6] using CARLA for camera sensors.

IFAG: IFAG will contribute to the execution block of the SAF by preparing the execution of the allocated test cases. The actual execution will be done using the described virtual testing setup which eventually enables to demonstrate that the execution block of the SAF is able to handle EuroNCAP scenarios as well.

RESA/RSA: RESA will prepare the execution of the allocated test cases on the AD prototype. This proves if all tests could be executed and if the needed signals can be obtained for further test evaluation.

SISW: Siemens plans to execute allocated test cases on its slightly modified V&V virtual simulation framework defined in D4.4 [6]. The additional changes include a perception pipeline to meet test case requirements.

VIF: VIF will prepare the execution of the allocated test cases. The execution will be done by using a customised version of the harmonised V&V simulation framework described in D4.4 [6].

Test Evaluate

CVC: As all the testing scenarios are done virtually, we will use a collision and route completion metric to decide the pass/fail criteria of the scenarios. Other metrics provided by the SAF can be computed to compare.

RESA/RSA: RESA will review the evaluation process based on the obtained parameters on the vehicle and receive input from the external requirements. In this use case, a good metric for analysing a perception system could be if a specific obstacle has been properly classified/detected and if the distance/ speed to the obstacle is also correct.

Decide

IFAG: IFAG will contribute to the Decide block of the SAF by ensuring that the pass/fail criteria of this block include the criteria from EuroNCAP, demonstrating that this block of capable of handling these types of scenarios as well.

RESA/RSA: RESA will evaluate if at that stage there is all the needed information from the "Coverage" and "Test evaluate" blocks to determine if the perception system is safe or not. Those metrics could be accuracy/ precision/ recall and average tracking errors.

SISW: Following the execution of virtual test cases, some metrics such as collision or time-to-collision will be used to evaluate the severity of scenarios and group them as safe or unsafe. Additionally, unsafe scenarios may be categorized under failed SAF subsystem, including perception or control blocks.

3.2 Use case 1.2: Urban AD validation – Connected perception testing

In Use Case 1.2, the SUNRISE SAF is tested for cooperative perception and decision-making in urban intersection scenarios. The system under test is a cooperative Adaptive Cruise Control (ACC) that is expected to improve with enhanced perception obtained through Vehicle to Everything (V2X) connectivity.

There are four functional scenarios defined in T7.1. Three of these scenarios involve improving vehicle behaviour when approaching a traffic light and receiving traffic light phases, timings, and map information. The scenarios cover three types of unexpected events: a baseline scenario with no unexpected event, a violation by a distracted pedestrian, and a reset of phases due to a pedestrian call. The fourth scenario deals with a red-light violation by a crossing car, and in this case, V2V information complements the traffic light connection.

This Use Case involves the following partners:

- UNITN will run simulations in the IPG CarMaker environment with a model of the systems under test. Traffic light phases and timings can be programmed, and V2V communication is simulated via SimNet.

- ika cooperates with UNITN as shown in the table below.

- VEDECOM. As part of this experiment, our aim at VEDECOM is to validate the value of SUNRISE SAF, integrated into the infrastructure, for improving the perception and decision-making capabilities of an autonomous vehicle in cooperative scenarios. This experiment focuses specifically on validating the longitudinal control of an autonomous vehicle in the context of intelligent traffic light management and intersection crossing. VEDECOM. The experiment is divided into several stages. First, critical scenarios are identified using a functional simulator developed in Matlab/Simulink for autonomous vehicle control. This simulator focuses on the analysis of traffic light management and intersection crossing scenarios. These critical scenarios are then tested using the SUNRISE on-board simulator, which enables the evaluation of complex and dangerous conditions, such as collisions at intersections or events triggered by traffic light phase changes, depending on vehicle position or speed.

3.2.1 Setup

3.2.1.1 Virtual testing

Virtual testing is carried out by UNITN and VED. UNITN will use the IPG CarMaker environment described in SUNRISE deliverable D4.3 [9], section 2 (2.1.4 for test case manager, 2.2.4 for the environment, 2.3.5 for the test vehicle, 2.4.2 for the traffic agents, 2.5.1 for connectivity).

The use of simulation tools enables VED to reproduce a wide range of scenarios that would be difficult, time-consuming, or even dangerous to test under real-life conditions. This approach not only improves the efficiency of the testing process, but also provides a controlled environment for evaluating the performance of AD systems, the infrastructure simulator, and the VEDECOM simulator adapted for SUNRISE project under a variety of conditions.

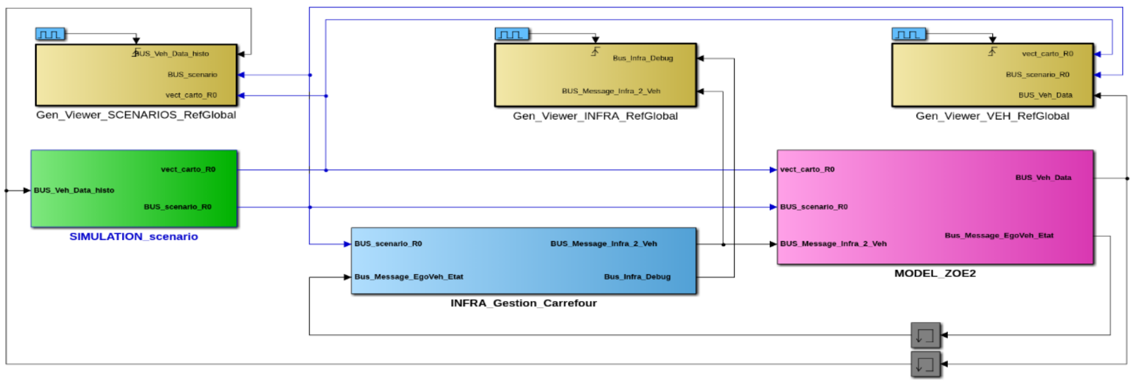


Figure 16: VEDECOM's AD simulator architecture

The AD simulator is designed at the highest level as follows:

<u>Simulation_Scenario:</u>

This component consists of two modules: the scenario generator and the viewer. The scenario generator creates and manages virtual objects and obstacles within the scene, while the viewer provides a comprehensive display of the overall scene, allowing for detailed visualization and analysis of the simulated scenarios.

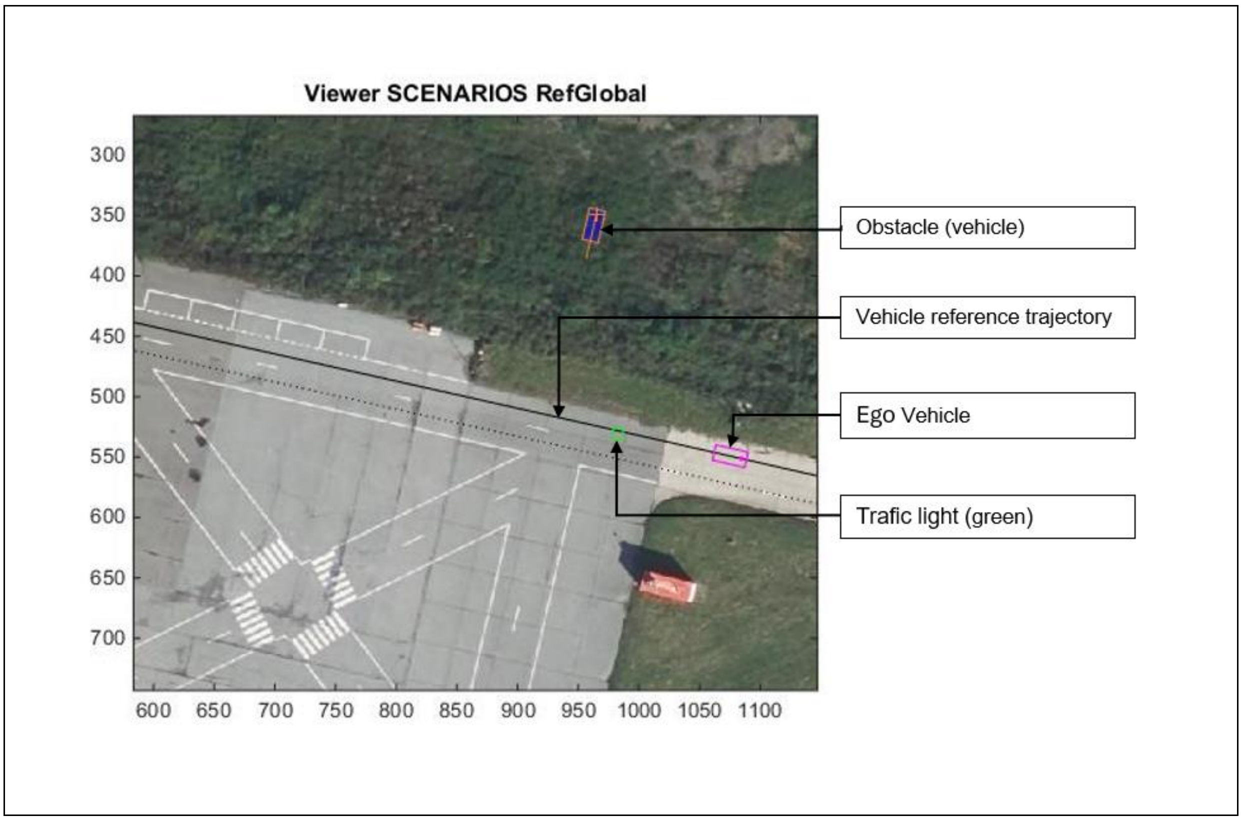


Figure 17: Scenario simulation viewer in the global reference repository

In Figure 17 the scenario simulator viewer defines the obstacle with an orange bounding box, indicating that it is a vehicle type, with a blue interior colour. The trajectory of the ego vehicle, shown in magenta, is represented by a continuous black line. The green circle represents the traffic light, which is in the green state.

Infra_Gestion_Carrefour (Infra_Management_Intersection_AD):

This module interprets the scene and makes decisions from the infrastructure's perspective for the automated vehicle. It consists of a module and a viewer. It is triggered by the reception of perception data, allowing for the management of complex intersection scenarios.

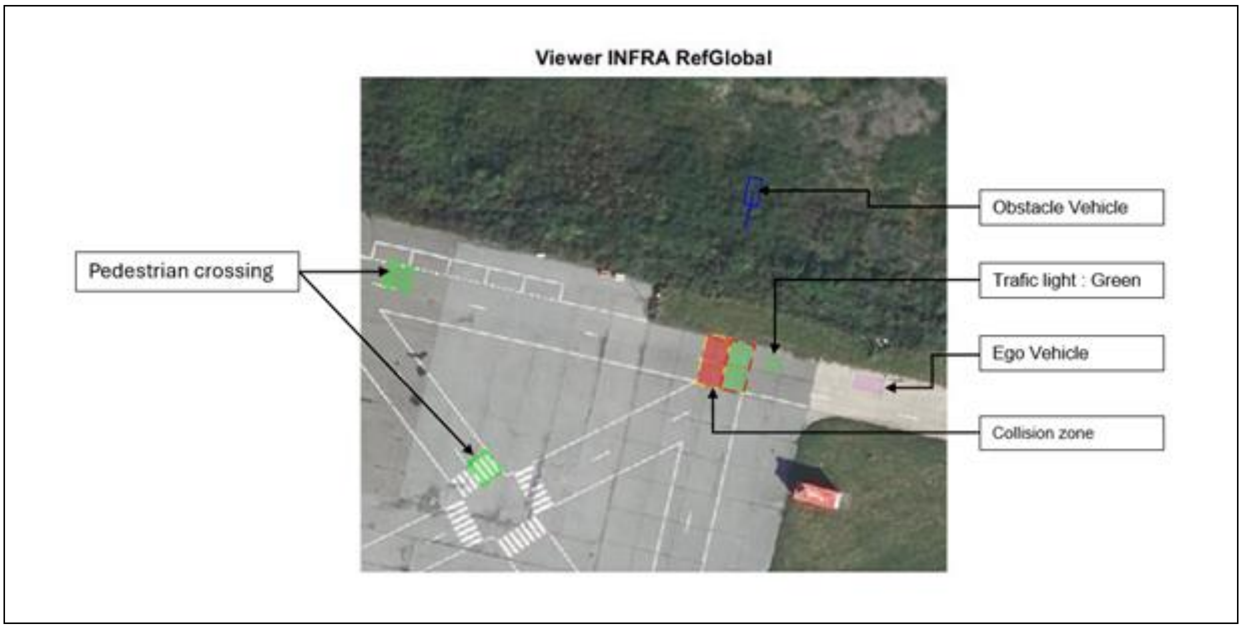


Figure 18: infrastructure viewer in the global reference repository

In Figure 18, the infrastructure viewer displays the collision zone, The yellow bounding box indicates a priority collision zone (traffic yield). If the colour is red, it means the yield zone presents a risk, while green signifies that there is no risk. The ego vehicle is also represented from the infrastructure perspective, but it is not considered an obstacle for itself and is shown as transparent. The traffic light is green, and the obstacle vehicle is depicted in blue from the infrastructure’s perspective. Additionally, pedestrian crossings are represented in green, as they are prioritized and unoccupied in this simulation.

Model_ZOE (AD_Vehicle_Model):

This level includes the vehicle and the AD simulation model and its associated viewer. The model is responsible for managing the vehicle's movements, including acceleration, deceleration, and braking. It integrates messages received from the infrastructure and responds appropriately to various situations, providing a comprehensive simulation of the vehicle's behaviour in different scenarios.

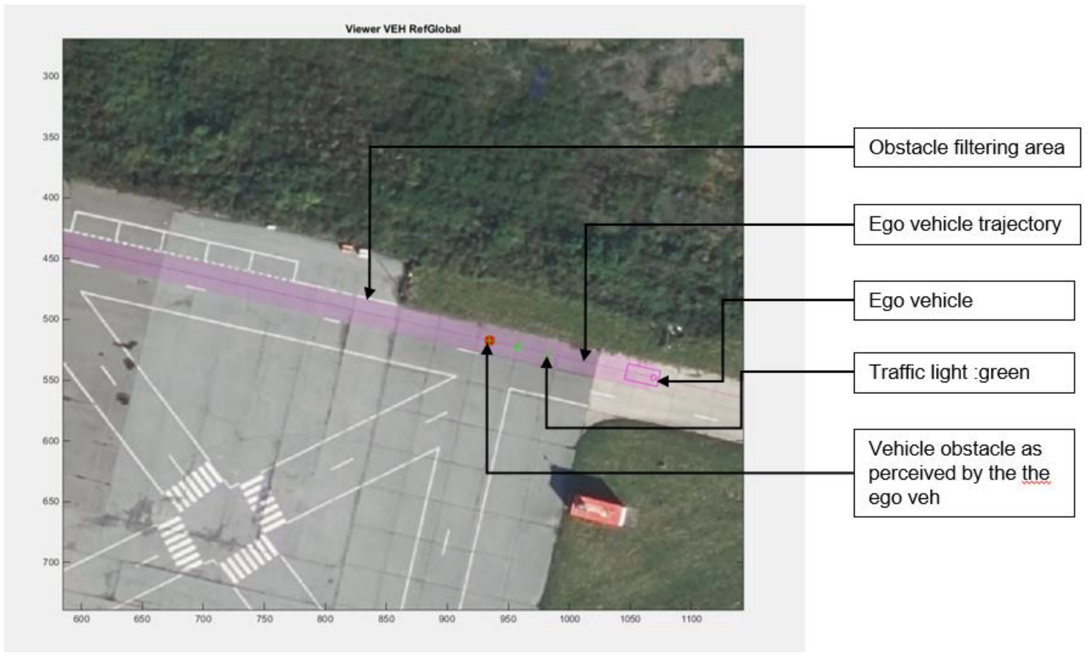


Figure 19: Vehicle viewer in the global reference repository

In Figure 19, from the vehicle's perspective, the ego vehicle's trajectory is visible along with an obstacle filtering zone. Among these obstacles, the green traffic light is detected. The obstacles perceived by the infrastructure, shown as bounding boxes, are represented in the ego vehicle's map as green circles, with each point indicating the closest part of the obstacle to the ego vehicle.

3.2.1.2 XiL testing

In this phase, VEDECOM will be concentrating on two distinct experiments. Testing is carried out on dedicated test tracks at Satory, Versailles, which are designed for testing and experimentation. These areas are secured, ensuring that there are no uncontrolled obstacles. Each experiment is designed to validate different aspects of the AD system. Although the system is already complete without the VEDECOM simulator adapted to the SUNRISE project, this additional tool plays a crucial role in qualifying the system by simulating virtual objects, including collision scenarios.

Experimentation 1: Traffic Signal Management

The first experiment centers on Traffic Signal Management. The complete system for this experiment consists of the infrastructure, connectivity, the vehicle, and the SUNRISE simulation (see Figure 20).

The core VEDECOM system consists of an automated vehicle that communicates with an infrastructure equipped with an intersection manager. This manager controls the traffic light status and sends this information to the automated vehicle. In turn, the vehicle transmits its data to the infrastructure manager, mainly including its position, speed, heading, and whether it is in autonomous mode or not.

The VEDECOM simulator adapted for SUNRISE project communicates with the infrastructure manager by sending the traffic light object and receiving vehicle data.

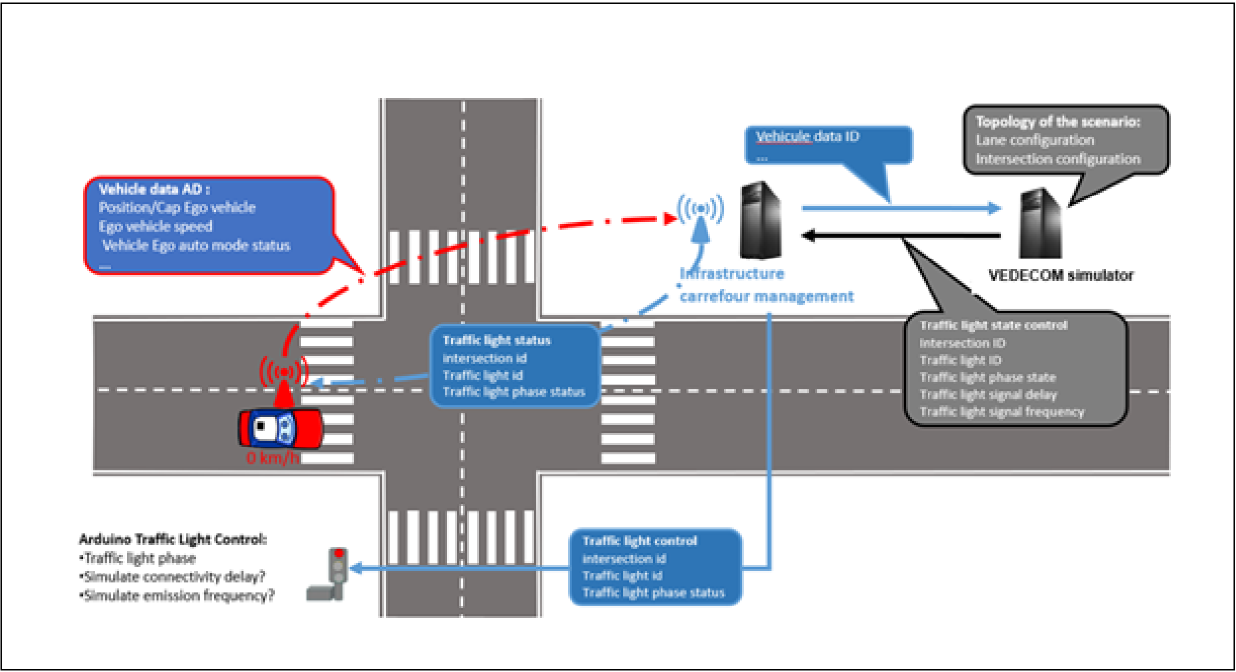

Vehicle data AD :
Position/Cap Ego vehicle
Ego vehicle speed
Vehicle Ego auto mode status
...
0 km/h
Traffic light status
intersection id
Traffic light id
Traffic light phase status
Vehicule data ID
...
Topology of the scenario:
Lane configuration
Intersection configuration
Infrastructure
carrefour management
VEDECOM simulator
Traffic light state control
Intersection ID
Traffic light ID
Traffic light phase state
Traffic light signal delay
Traffic light signal frequency
Arduino Traffic Light Control:
•Traffic light phase
•Simulate connectivity delay?
•Simulate emission frequency?
Traffic light control
intersection id
Traffic light id
Traffic light phase status

Figure 20: Signal light control

<u>Experimentation 2: Intersection Management</u>

The second experiment focuses on Intersection Management. Similarly, the complete system includes the infrastructure, connectivity, the vehicle, and the simulation tools (see Figure 21).

In this experiment focusing on intersection management, the automated vehicle sends its data (position, heading, speed, and autonomous mode status), to the intersection manager. This manager then forwards the information to the VEDECOM simulator adapted for SUNRISE project, which generates a list of virtual obstacles based on the received data. These obstacles are subsequently sent to the infrastructure manager, who communicates them to the automated vehicle along with the authorization or restriction for crossing the intersection.

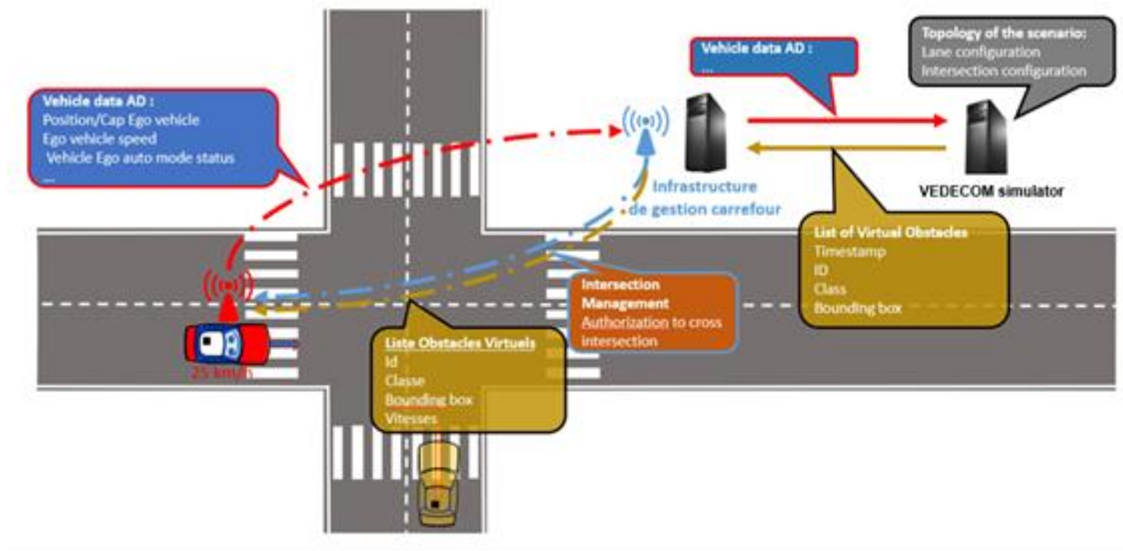


Figure 21: Intersection management

3.2.1.3 Proving ground testing

In the proving ground phase, validated scenarios and system components are taken from the simulation environments and are subjected to real-life conditions on a controlled test track.

Field testing involves several key elements:

- Real-time scenario execution
- Interaction between vehicle and infrastructure
- Safety and reliability testing

The current KPIs (Key Performance Indicators) that have been established are designed solely to evaluate the performance of the automated vehicle control system and connectivity, but they do not assess the Vedecom Simulation system.

Traffic light crossing KPI

The **traffic light crossing** KPI (Key Performance Indicators) assesses several performance indicators related to decision-making and vehicle control during a stop at a traffic light. It is structured into different sections.

Decision to Stop at Traffic Light:

Stop / No Stop Initiated and **Stop Triggered** evaluate whether the decision to stop the vehicle was correctly initiated.

Adaptation of Speed for Light Crossing examines whether a speed limitation was applied when crossing the light.

Robustness of the Decision measures the stability of the decision, with an indicator showing the number of state changes in the decision.

Reactivity of the Decision calculates the reaction time between the light state change and the activation of the stop planning, with thresholds defining what is acceptable or not.

Longitudinal Control Performance:

Vehicle Control Precision assesses the stopping distance relative to the light line, and the front bumper of the vehicle with precision thresholds ranging from very precise to unacceptable.

Vehicle Control Reactivity measures the delay between the light state change and the vehicle's reaction, with thresholds for the speed of response (from no delay to being unacceptable).

Longitudinal Control Comfort:

Average Comfort measures the average acceleration/deceleration during the stop/restart phase, with comfort levels depending on the intensity of the acceleration.

Momentary Discomfort focuses on acceleration, deceleration and jerk peaks, assessing the impact of sudden variations in acceleration and deceleration.

<u>Intersection crossing KPI</u>

The **Intersection Crossing** KPI table assesses several performance indicators related to decision-making and vehicle control when crossing an intersection. It is divided into different sections.

Decision to Stop at Intersection:

Stop / No Stop Initiated evaluates whether the decision to stop the vehicle at the intersection has been correctly initiated, with the criteria being a stop if there is a priority obstacle or a high collision risk.

Speed Adjustment for Intersection Crossing examines whether the speed limitation has been applied when crossing the intersection.

Robustness of the Decision measures the stability of the decision, indicated by the number of state changes.

Reactivity of the Decision calculates the reaction time between detecting a priority obstacle and the stop planning (collision zone generation), with specific thresholds.

Longitudinal Control Performance:

Vehicle Control Precision assesses the stopping distance relative to the target distance, considering a safety buffer from the collision zone. A short stopping distance indicates high precision, while a larger deviation suggests varying levels of acceptability or unacceptability.

Vehicle Control Reactivity measures the delay between the appearance of a collision zone and the vehicle's response. Faster response times are preferred, with increasing delays indicating reduced performance, ranging from acceptable to unacceptable.

Collision with Obstacle checks whether a collision has occurred, with the target criterion being no collisions.

Longitudinal Control Comfort:

Average Comfort during the stop sequence is evaluated by measuring the average acceleration and deceleration during the vehicle's stop or slow down phase. Comfort levels are to be defined based on the scenario, with normal ranges aggressive, and any acceleration indicating an emergency stop.

Momentary Discomfort focuses on peaks of deceleration, acceleration and jerk during the stop sequence, assessing the maximum acceleration/deceleration as well as the jerk (rate of change in acceleration). Comfort thresholds are to be defined based on the scenario, with reference values of normal comfort to aggressive behaviour.

3.2.2 Covered aspects of the SAF

The ODD and the four functional scenarios are defined in SUNRISE D7.1 [1]. These are used to generate logical scenarios, i.e., parametrisations. For this use case, the parameters and the ranges are not obtained for the SAF and are, instead defined internally to WP7. In addition, the KPIs are also defined by WP7.

The testing of the SAF begins at the concretise step and ends at the simulation validation step. The table below shows the contributions to the SAF block validation per partner for UC1.2.

Table 4: Contributions to the SAF validation per Partner in UC1.2

Partner/ Block	ika	UNITN	VED
SUNRISE DF			
Query & Concretise	x	x	
Allocate	x	x	x
Execute	x	x	x

Test Evaluate	x	x	x
Coverage	x*	x	x
Decide		x	x

*only parameter space coverage

Query & Concretise

ika: ika provides concrete scenarios for simulation based on the methodology developed in WP3. For the case generation, the query process is a required input and will be validated against the validation criteria defined in chapter 2 for the logical scenarios defined in the use case. The sampling methodology will be validated based on the coverage of the parameter space by the KPI's defined in the use case.

UNITN: UNITN defines the logical scenario, and IKA provides the sampling according to the SAF block.

Allocate

The UC will use the environment allocation from the SAF. Initial allocation will be on simulations. However, in a second run, a few cases will be re-allocated to physical testing. The efficiency of this re-allocation (in particular whether the most relevant cases were selected) may be tested with the above criteria.

ika: Ika provides an allocation strategy based on the methodology developed in WP3. The allocation strategy can be validated by comparing the results of the simulation and physical testing against the output of the allocation strategy.

UNITN: UNITN conducts the cases that are assigned to simulations. By evaluating whether the simulations successfully generated accurate evaluation metrics, one can determine whether the simulation test instances were appropriate or if the allocation block should have indicated test instances with higher fidelity.

VED: VED has defined a list of cases based on three parameters: vehicle speed, the distances to the traffic light, and the state of the light. An essential variable is added to this: the occupancy of the crossing area.

Execute

ika: ika provides physical testing for the use case on a proving ground and validates the execution block for physical testing.

UNITN: UNITN runs the simulations and reports whether each concrete scenario was correctly executed or if it was not completed and why.

VED: VED executes these concrete scenarios on the simulation model in the loop and with a car on VED's Proving ground. We must mention that VED simulator and the vehicle have exactly the same core functions.

Coverage

ika: ika determines coverage of the cases in the parameter space, necessary for the methodology. By comparing against results from simulation and real-world testing we validate that the surrogate model from which the generated test cases are derived provides an approximation of the underlying distributions of the KPI's over the parameter space. This only validates part of the "Coverage" block since scenario coverage cannot be determined.

UNITN: IKA gives the SAF coverage that UNITN cross-checks with one of the following methods: a) new independent test samples or (either in the parameter space or with new parameters that were not considered in the logical scenario, b) with a surrogate ML model of the test results across the logical scenario (as in UNITN's paper under review).

VED: VED will determine the coverage of the test cases.

Test Evaluate

ika: ika conducts test evaluation necessary for the methodology and uses the validation criteria to validate the physical tests that ika performs.

UNITN: UNITN returns the performance indicators per concrete scenario. IKA evaluates the KPIs according to the SAF. UNITN compares these evaluations with internal evaluations.

VED: VEDECOM defined a set of KPIS in section 3.2.1.3. Once the execution is done the KPIs are computed for every single test case and evaluated for both simulation and real car and compared.

Decide

UNITN: UNITN will cross-check the SAF decision with one of the methods mentioned above.

3.2.3 Deviations from D7.1

After CRF withdrawal, the systems under test are no longer the same. Simulations carried out by UNITN are based on two or three different and partially reconfigurable systems with no exact physical counterpart. Physical tests by ika use a different function. Only VED has the same system for virtual and physical tests.

Nonetheless, at some higher functional level, ACC systems must be acceptable to human drivers. This means that the longitudinal acceleration is limited (except in emergency conditions) and that longitudinal control is smooth (jerk is bounded).

Hence, we can compare the results (simulations and physical tests) to naturalistic driving distribution for longitudinal accelerations and jerks.

3.3 Use case 1.3: Urban AD validation – Cooperative perception testing

In Use Case 1.3, the SUNRISE SAF is tested for cooperative perception and decision-making in urban intersection scenarios. The SUT is the perception AD subsystem of an urban chauffer ADS which is also capable of sending/receiving and processing (on-board or off-board e.g. employing a remote smart RSU node) rich V2X information data mainly consisting of object-level data perceived from other road users.

There are three functional scenarios defined from T7.1 for assessing the cooperative perception system performance in Use Case 1.3. The first is the “darting-out pedestrian” scenario (1.3 A), the second one is the “urban junction” scenario (1.3 B), and the third one is the “urban roundabout” scenario (1.3 C).

Use Case 1.3 involves the partners ICCS and VED.

ICCS will run simulations in a virtual environment, based on a CARLA-OpenCDA co-simulation environment that will be developed using the open-source ms-van3t framework, where V2V communication is realized by integrating ns-3 simulator. The SUT is an in-house Collective Perception module that fuses object-level information coming available from other agents which deploy single-agent perception and are capable of sending and receiving CP messages as those are standardized by ETSI CP service. ICCS has also foreseen the possibility of conducting a hybrid simulation study in which CARLA would be receiving input from external real agents operating in a controlled test track.

VED will run simulations in a custom environment created in SUMO and VEINS and carry out corresponding XiL tests on proving grounds.

3.3.1 Setup

3.3.1.1 Virtual testing

The use of simulation tools enables us to reproduce a wide range of scenarios through virtual testing that would be difficult, time-consuming, or even dangerous to test under real-life conditions. This approach not only improves the efficiency of the testing process but also provides a controlled environment for evaluating the SUNRISE SAF, with a SUT model, under various conditions.

In Use Case 1.3, different flavours of virtual testing are provided by ICCS and VED as initially described in D4.3 [9].

3.3.1.1.a Virtual testing carried out by VED

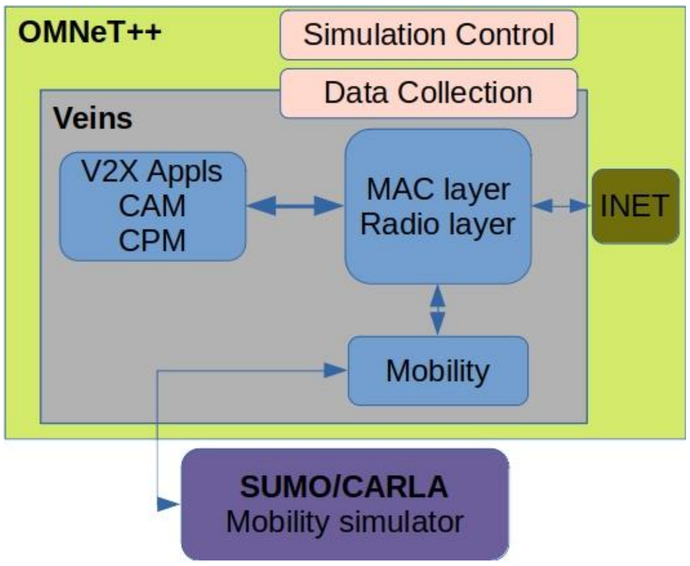


Figure 22: Virtual testing simulation framework carried out by VED for UC 1.3 (test scenarios 1.3 A and 1.3 B).

In this section, an overview of the virtual simulation framework carried out by VED is based on OmNet++, Veins, and SUMO/CARLA simulators. The virtual simulation framework carried out by VED is shown in Figure 22 which integrates several components to create a virtual environment for testing and validating scenarios 1.3 A and 1.3 B. Here's a breakdown of each component and its role:

- **OMNeT++**: Serves as the central simulation environment. It includes several key modules.
- **Veins**: A framework for V2X (Vehicle-to-Everything) connectivity, embedded within OMNeT++ and connected to other modules for simulating communication between vehicles.
    - **V2X Applications:** Handles the V2X applications based on V2X communications messages such as the CAM and CPM.
    - **CPM Implementation:** Cooperative Perception Messages (CPM) are implemented within Veins, allowing the simulation of -V2X- communication for enhanced situational awareness.
    - **Perception Model:** Veins includes a perception model that simulates how vehicles perceive their environment, contributing to the generation and reception of CPMs.
    - **Vehicle Control Algorithm on Reception of CPM:** A vehicle control algorithm is implemented to manage how vehicles respond when they receive CPMs, influencing their behaviour in the simulation.
    - **MAC and physical layer**: simulates the Medium Access Control (MAC) layer and physical radio layer of the V2X communication.
    - **Mobility**: Handles the movement and traffic dynamics of vehicles, connected to the mobility simulators SUMO and/or CARLA.
- **SUMO:** simulates the road network and traffic routing.
- **CARLA:** Responsible for scenario simulation, CARLA provides a realistic virtual environment where various driving scenarios are simulated. It interacts with Veins via gRPC connector.

- **INET**: A network simulation framework within OMNeT++ that interfaces with Veins providing network-related simulations.

Below it is described how the testing of scenario 1.3 A using the virtual simulation framework carried out by VED is executed. The procedure of simulation for scenario 1.3 B is similar.

#1. Setup the Scenario in CARLA

- **Design the Environment:** In CARLA, create the road scenario where the ego vehicle will operate. This setup should include the road segment where the darting-out pedestrian event will occur. Position parked vehicles (e.g., a bus) that will occlude the pedestrian from the ego vehicle's view.
- **Place the Ego Vehicle:** Position the ego vehicle (CAV) on the road, approaching the area where the pedestrian will appear.
- **Position the Pedestrian:** Place the pedestrian behind the occluding vehicle, ready to dart into the path of the ego vehicle.

#2. Traffic and Mobility Simulation in SUMO

- **Traffic Flow:** Use SUMO to simulate traffic flow, including the movement of the ego vehicle and any other vehicles, such as the opposing vehicle that might detect the pedestrian.
- **Pedestrian Dynamics:** Configure the pedestrian's movement in SUMO, ensuring it synchronizes with the ego vehicle's approach.

#3. V2X Communication Simulation in Veins

- **V2X Message Generation:** In Veins, configure the generation of V2X messages, specifically CAMs and CPMs. For instance:
- **Test Case 1:** No CPMs are generated; the ego vehicle relies solely on its sensors.
- **Test Case 2:** The opposing vehicle detects the pedestrian and generates a CPM containing pedestrian information.
- **Test Cases 3 & 4:** Similar CPMs are generated, but they convey information that the pedestrian is either not on the crosswalk or has already crossed.
- **MAC and Physical Layer Simulation:** Ensure the V2X messages are transmitted over the simulated network, considering the MAC and physical layers.

#4. Network Simulation in INET

- **Simulate Network Conditions:** Use INET within OMNeT++ to simulate the network conditions, such as delays or packet loss, which might affect the transmission of CPMs from the opposing vehicle to the ego vehicle.

#5. Scenario Management and Synchronization in OMNeT++

- **Integrate Components:** OMNeT++ will act as the central hub, integrating CARLA, SUMO, Veins, and INET. This ensures that all components operate in sync, with the traffic simulation, V2X communication, and network conditions all being managed together.

- **Run the Simulations:** For instance:
- **Test Case 1:** The ego vehicle operates without any CPMs and collides with the pedestrian.
- **Test Case 2:** The opposing vehicle sends a CPM after detecting the pedestrian, allowing the ego vehicle to brake and avoid the collision.
- **Test Cases 3 & 4:** The ego vehicle receives CPMs that indicate the pedestrian's location, allowing it to decide whether to stop or continue safely.

#6. Data Collection and Coverage Analysis

- **Collect Performance Data:** Gather data on the ego vehicle's behaviour, including reaction times, braking actions, and whether the collision was avoided.
- **Analyse the Impact of V2X:** Compare the scenarios with and without CPMs to evaluate the effectiveness of V2X communication in preventing collisions.

By following these steps, the selected test scenarios (1.3 A and 1.3 B) of UC 1.3 can be simulated and analysed with respect to improving the safety of VRUs, highlighting the importance of co-simulation safety assurance frameworks supporting testing of V2X communication together with AD systems under test i.

3.3.1.1.b Virtual testing carried out by ICCS

ICCS will use CARLA simulator and CARLA-network co-simulation environment as described in the deliverable D4.3 [9].

The ICCS simulation framework, as specified in SUNRISE deliverable D4.4 [6] (see Figure 23), serves as the foundation for conducting virtual tests, providing a robust and reliable platform for validating the safety of SUT in Use Case 1.3 through the SUNRISE SAF. This comprehensive framework is comprised of two primary components:

- CARLA simulator: A Python API functionality is used to introduce dynamic elements such as additional traffic (comprising vehicles and pedestrians), dynamically manage traffic lights, and manipulate environmental conditions such as weather and sun position based on the time of day. In this context, versatile and adaptive test environments are created, and thus, a wide range of driving scenarios with varying levels of complexity and realism are simulated.
- CARLA-network co-simulation environment building upon the open source MsVan3t (/ADD REF) framework: It uses CARLA simulator for mobility and sensor perception simulation and ns3 simulator for network simulation. This extension is able to extract not only localization information from CARLA simulator but also perception information from the Local Dynamic Map (LDM) module. The developed client module on ns-3 queries the information to use it for the mobility of each of the ns-3 simulated nodes and to update the LDM module with all perception data sent over the simulated vehicular network.

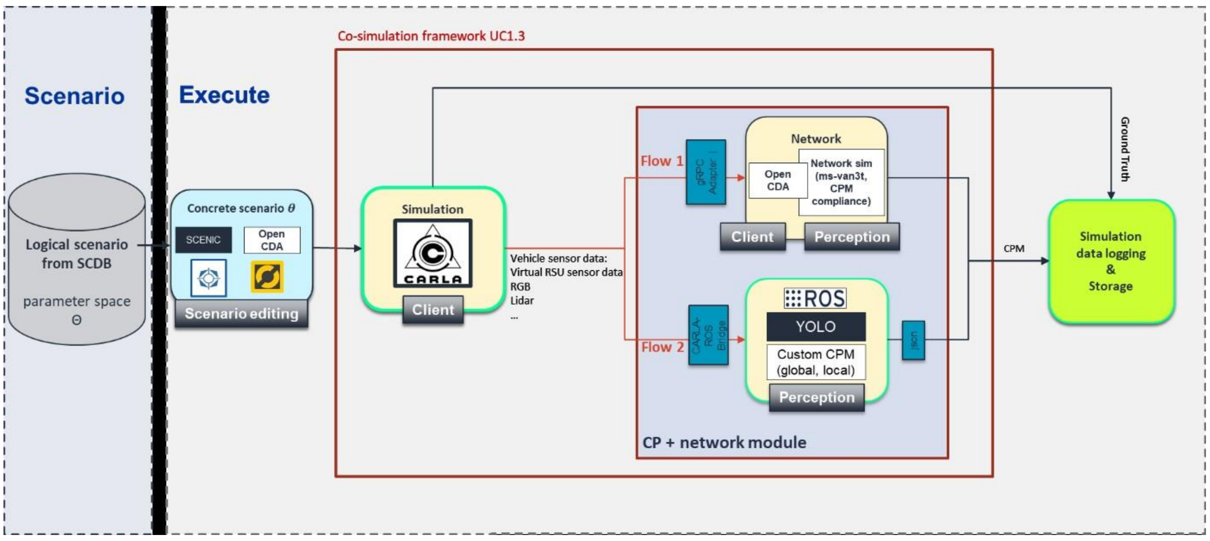


Figure 23: Virtual co-simulation framework carried out by ICCS for UC1.3.

The virtual co-simulation framework carried out by ICCS includes two possible simulation flows (see Figure 23). The first one (Flow 1) is based on OpenCDA with gRPC interface and ms-van3t where the ETSI-compliant CPM exchange and the fusion of object-level CPM information from different local perception providers is performed by the ms-van3t. The second one (Flow 2), more-simple, is based on YOLO and an ICCS CP module integrated via ROS2 and CARLA-ROS-bridge for custom local and global CPM generation where the network simulator is not used at all and message exchange is emulated. In either case, the produced CPM along with the raw sensor data (CARLA output) and the ground truth information available in CARLA will be logged and stored in order to support offline evaluation of the CP module.

The ICCS virtual co-simulation framework will be executed as described previously starting from a concrete scenario drawn from the SCDB logical scenarios. It will be evaluated through data replay based on the information that is logged during the course of each simulation execution round. The evaluation will be performed in two possible ways (see Figure 24):

Evaluation of the collective perception (CP) module

In this case, only the CP module will be evaluated based on KPIs related to perception such as accuracy of object detection, surface overlap between the output probabilistic occupancy grid and the ground truth bird-eye view provided by CARLA. No planning and control modules will be employed.

Evaluation of the full AD stack
In this case, a planner and controller will receive the output of the CP module and based on this will determine the dynamic behaviour of the ego vehicle. The KPIs here are related to safety metrics such as time to collision etc.

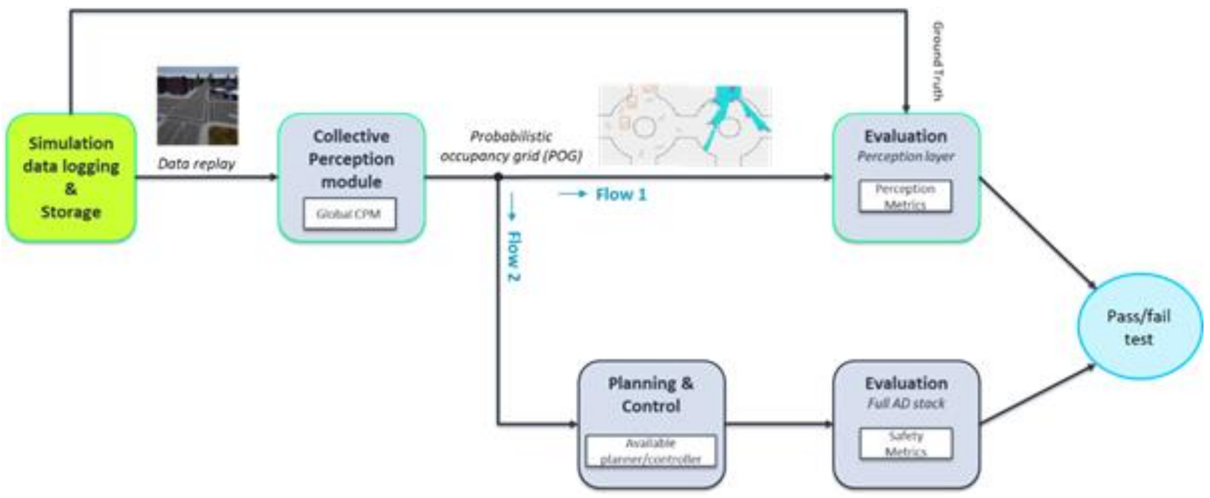


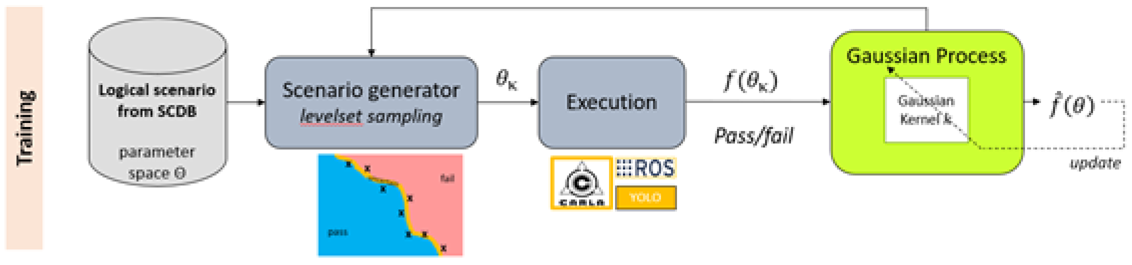
Figure 24: ICCS evaluation pipeline with data replay

In both cases, the evaluation metrics will be applied in conjunction with respective thresholds that will determine whether the AD system passes or fails the specific safety test case. The existence of such thresholds and pass/fail criteria for the SUT, in this case, the CP system is very important for the scenario generation methodology and it will be described in detail below.

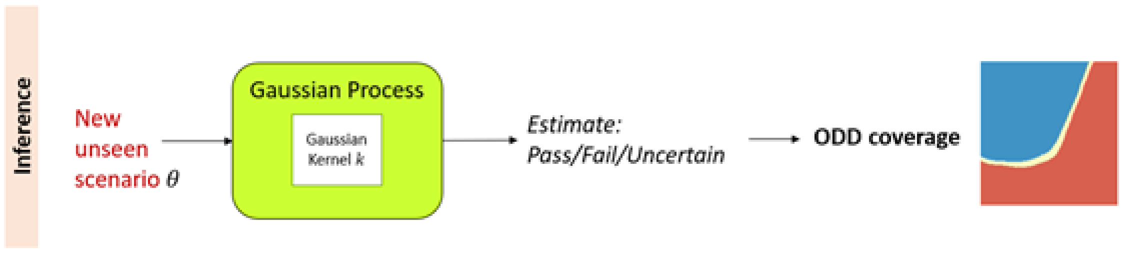
Special aspects: Joint scenario generation and testing approach

A probabilistic method based on Gaussian Processes (GP) will be employed for the generation of critical scenarios for perception testing and will be evaluated by UC 1.3. This is a pre-release of what is to be included in the relative deliverable D3.4. The advantage of this method is that it can provide a method for scenario space exploration and scenario coverage.

The method is described in Figure 25. The ultimate goal is to train a GP to estimate whether a concrete scenario will pass or fail a task without the need to execute it in simulation. Starting from a logical scenario, which is described by a parameter space, i.e. a set of scenario parameters and their ranges (for continuous parameters) or set of values (for discrete parameters), concrete scenarios are extracted, simulated, and evaluated as pass or fail based on a pre-defined set of metrics related to the task (e.g. perception-based metrics). The extraction of the concrete scenarios is performed by a level-set sampling algorithm which selects scenarios close to the pass/fail boundary. These scenarios are considered the most critical ones to obtain as i) their outcome is uncertain, and ii) they define the boundary between pass and fail regions that need to be learned. The selected scenarios and their respective outcomes are iteratively fed into the GP to update its parameters (see Figure 25a). At the end of this process, upon convergence of the GP, it is possible to use the trained GP as a predictor of unseen scenario outcomes without simulation. This process is much faster than the simulation, hence, it is possible to query almost any concrete scenario of the logical scenario space and ultimately acquire coverage estimation (see Figure 25b). Moreover, it is possible to generate more critical scenarios (i.e. on the boundary of the pass/fail space or on the 'fail' region of the pass/fail space) to assess through simulation if this is necessary.



(a)



(b)

Figure 25: Critical (levels-et) scenario extraction and simulation and GP training (a) and GP querying to estimate outcomes of unseen scenarios and assess coverage (b)

3.3.1.2 XiL Testing

XiL or hybrid testing is employed as a complementary way of testing if real HW components can be used in order to get closer to real-deployment conditions. In this UC, the interesting real-deployment condition is a real agent in a test trackable to send CPMs as well as to receive CPMs created by the agents in the virtual environment (V2I or V2N connectivity of the agent is assumed). Again, ICCS and VED provide two different hybrid testing environments and focus on different selected UCs.

3.3.1.2.a XiL testing carried out by VED

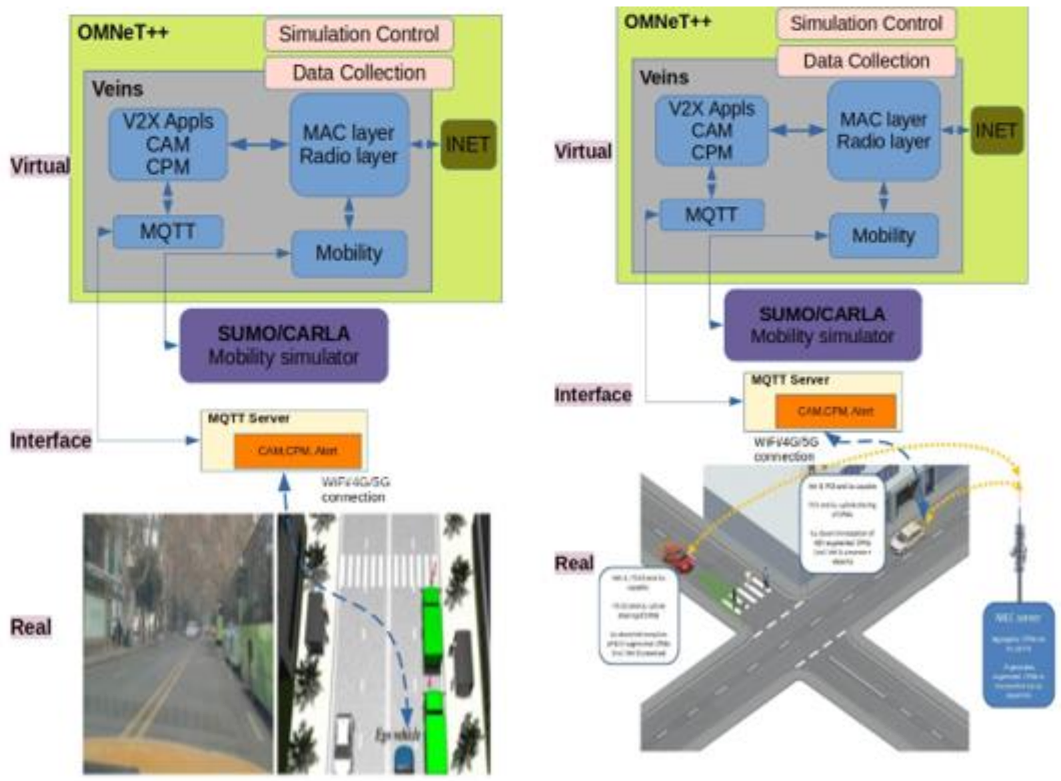


Figure 26: XiL simulation framework diagram for scenarios 1.3 A (left) and 1.3 B (right)

The simulation frameworks for test scenarios 1.3 A and 1.3 B are shown in Figure 26. This framework integrates virtual and real-world environments through a well-defined interface. Below the three components of this framework are described.

Virtual Environment: The virtual environment of XiL simulation is the same as the virtual simulation frame except that it includes an extra component of MQTT client. Veins incorporate an MQTT client to publish and subscribe to CPM information, facilitating communication between the virtual and real environments.

Interface:

MQTT Server: The interface between the virtual and real worlds is facilitated by an MQTT server. MQTT is a lightweight messaging protocol, ideal for devices with limited bandwidth.

CAM, CPM, Alert: The server handles various types of messages:

CAM (Cooperative Awareness Message): Messages used in vehicular networks to share information about vehicle positions and movements.

CPM (Cooperative Perception Message): Messages that share sensor data or perceptions between vehicles to enhance situational awareness.

Alert: Messages that may warn about potential hazards or urgent situations.

Real Environment:

Real-World Vehicles and Scenarios: The real environment, representing actual traffic scenarios, interacts with the virtual simulation through the MQTT server. The image shows both a real camera feed and a corresponding simulation, highlighting how the ego vehicle (the vehicle under test) interacts with other vehicles and infrastructure.

Wireless Communication (WiFi / 4G / 5G): The real-world vehicles are connected to the virtual environment through wireless communication networks such as WiFi, 4G, or 5G.

Below the XiL setup is described and how scenario 1.3 A is simulated using the above XiL framework. The procedure is similar for scenario 1.3 B. To simulate 1.3 A test scenarios using the XiL simulation framework, the process involves integrating the virtual environment with real-world elements through MQTT-based communication. Here's how each test case can be simulated:

#1. Simulation Setup:

Virtual Environment Configuration:

- **Intersection and Scenario Design:** Begin by setting up the same virtual environment in CARLA as described in the original virtual simulation framework. This includes modelling the darting-out pedestrian scenarios (1.3 A).
- **Vehicle Initialization:** The ego vehicle and other relevant vehicles are initialized in CARLA with predefined positions and movement trajectories. These vehicles are equipped with sensors and communication capabilities.
- **MQTT Client Integration:** In the Veins module within OMNeT++, incorporate the MQTT client, which will handle the communication of CAMs and CPMs between the virtual and real environments.

Interface Configuration:

- **MQTT Server Setup:** The MQTT server acts as a bridge between the virtual environment (Veins within OMNeT++) and the real-world environment. It manages the transmission of CAM, CPM, and Alert messages. The server ensures that real-world data (e.g., real-time sensor data from actual vehicles) can influence the virtual environment and vice versa.

Real Environment Configuration:

- **Real-World Vehicle Integration:** The real-world ego vehicle is equipped with the necessary communication hardware to connect to the virtual environment through the MQTT server. These vehicles transmit and receive CAMs, CPMs, and Alerts through wireless networks (e.g., WiFi, 4G, 5G).

#2. V2X Communication and MQTT Messaging:

Real-Time CPM Exchange:

- **Test Case 1 - Collision Scenario:** In this scenario, the ego vehicle operates without receiving CPMs from other vehicles. This means that the ego vehicle must rely solely on its onboard sensors (simulated in CARLA) to detect the pedestrian. The absence of CPMs simulates a situation where the pedestrian is occluded, leading to a collision. The MQTT client does not publish or subscribe to CPMs, reflecting a scenario where no additional data is available from other vehicles or infrastructure.
- **Test Case 2 - Critical Scenario:** Here, the ego vehicle receives CPMs from a vehicle in the opposite direction through the MQTT server. This data indicates the presence of a pedestrian, which is beyond the direct line of sight of the ego vehicle. The MQTT client in Veins subscribes to the CPM topic, allowing the ego vehicle to receive this crucial information in real-time, enabling it to slow down or stop in time to avoid collision.
- **Test Case 3 - Safe Scenario (Pedestrian Not Engaged on Crosswalk):** In this scenario, the ego vehicle receives CPMs indicating that the pedestrian is not engaged in crossing but is nearby. The MQTT client subscribes to these messages, allowing the ego vehicle to assess that it is safe to proceed without taking evasive action.
- **Test Case 4 - Safe Scenario (Pedestrian Has Already Crossed):** Similar to the previous case, the ego vehicle receives a CPM through the MQTT client indicating that the pedestrian has already crossed. The ego vehicle uses this information to continue its path without stopping, facilitated by real-time communication through the MQTT server.

#3. Scenario Execution:

Real-World Interaction:

- **Test Case 1 Execution:** The ego vehicle fails to detect the pedestrian due to a lack of CPM data from other vehicles, resulting in a simulated collision. This scenario tests the dangers of relying solely on local sensors in the absence of cooperative communication.
- **Test Case 2 Execution:** The ego vehicle successfully avoids a collision by responding to the CPM received from the MQTT server, illustrating the importance of extended situational awareness.
- **Test Case 3 and 4 Execution:** The ego vehicle evaluates the CPMs received and makes safe navigation decisions, highlighting how cooperative perception can enhance decision-making even in non-critical situations.

#4. Data Collection and Analysis:

Post-Simulation Analysis:

- **Data Collection:** After each test case, data on reaction times, decision-making accuracy, and collision avoidance success is collected from both the virtual and real environments.

- **KPI Evaluation:** The simulation results are analysed to assess the effectiveness of the MQTT-based communication in enhancing the ego vehicle's ability to avoid collisions or navigate safely.
- **Comparative Analysis:** Compare scenarios with and without CPM data to evaluate the added value of real-time cooperative perception in preventing accidents.

5. Simulation Duration and Repetitions:

- **Repetition for Robustness:** To ensure the validity of the results, each test scenario is repeated under varying conditions, such as different pedestrian speeds or varying wireless communication latencies.
- **Simulation Length:** Each simulation is run for a sufficient duration to capture the full sequence of events, from the pedestrian's appearance to the vehicle's response and the final outcome.

3.3.1.2.b XiL testing carried out by ICCS

ICCS has also foreseen the possibility of conducting a hybrid simulation study in which case CARLA would be receiving input from external real agents as shown in Figure 24. In this context, ICCS will conduct hybrid testing by utilizing a real agent, i.e. ICCS prototype vehicle at the ICCS premises in Athens, Greece (NTUA campus), or a pedestrian carrying a connectivity device. The real agent will be able to exchange CPMs and will be connected to the ICCS virtual co-simulation environment in real-time. For safety and testing purposes, this will be an isolated area, equipped with all the necessary elements to conduct the tests and implement an ETSI-compliant V2X communication with both real HW and emulated messages. These tests will mainly focus on executing the darting-out pedestrian test scenario (1.3 A).

Note: The hybrid setup wants to explore whether new SAF requirements are generated when real CPM exchange is tested. As such, this experiment will focus on the SAF “Execute” and “Test evaluate” blocks and not on the rest of the SAF components that are already evaluated in the virtual setup.

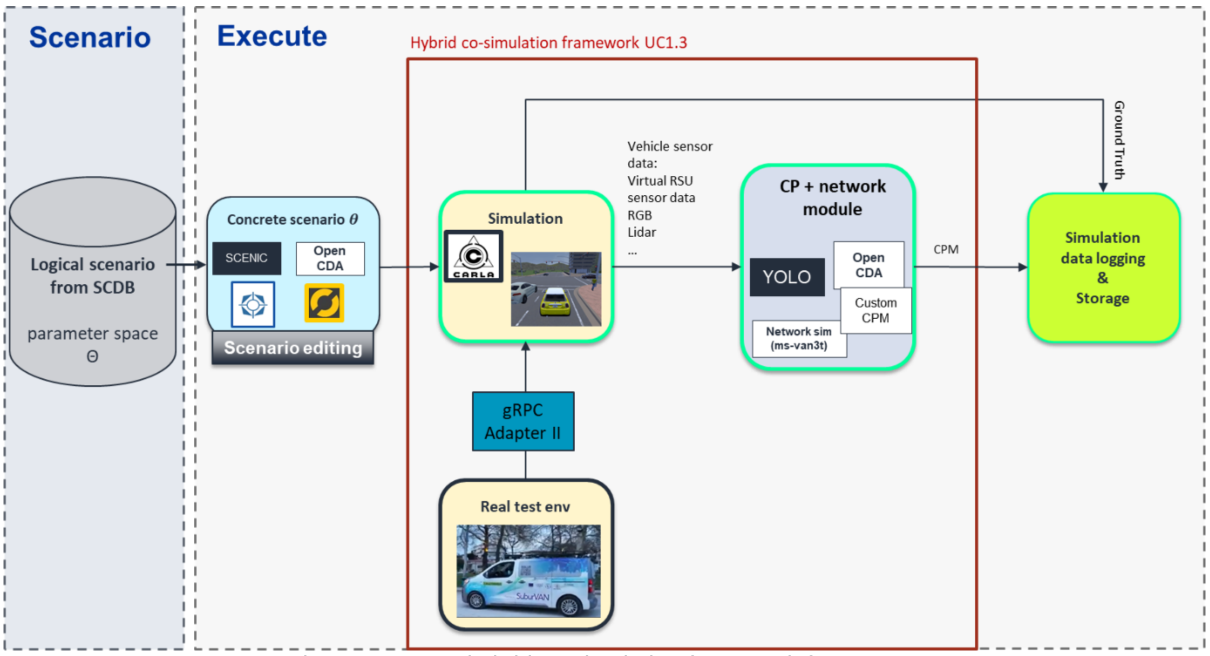


Figure 27: ICCS hybrid co-simulation framework for UC 1.3.

3.3.1.3 Proving ground testing

Additionally, VED will carry out vehicle-in-the-loop testing for test scenarios 1.3 A and 1.3 B on the proving ground test track of VED in Versailles, France. This isolated test track is equipped with road markings, pedestrian crossings, and all the necessary elements for conducting the tests.

3.3.2 Covered aspects of the SAF

The ODD and the three functional scenarios for use case 1.3 are defined in SUNRISE deliverable D7.1 [1]. These are used for generating logical scenarios. The parameters and the ranges are defined internally by WP7. In addition, the KPIs are also defined by WP7.

The UC will cover the steps from the definition of logical scenarios until simulation validation. The table below shows the contributions to the SAF block validation per partner for UC1.3.

Table 5: Contributions to the SAF validation per Partner in UC1.3

Partner/ Block	ICCS	VED
SUNRISE DF		
Query & Concretise	x	x
Allocate	x	x

Execute	x	x
Test Evaluate	x	x
Coverage	x	x
Decide	x	x

Query & Concretise

The work is based on a few selected hand-crafted logical scenarios suitable for Collective Perception Testing as described in D7.1 [1]. No need to query the SUNRISE DF. ICCS will create its own sets of concrete scenarios under one logical scenario using methods from T3.3 (smart concretization adhering to pass/fail uncertainty criteria) and using a custom scenario representation format.

Note: In order to feed the msVan3t co-simulation framework with scenario variations, some adaptation in its scenario editing library will be needed (yaml files dynamic configuration).

ICCS: ICCS will create an automatic probabilistic process (based on the work in T3.3) to concretize the scenarios stemming from a logical scenario described by a given parameter space with defined parameter ranges. The concretization will be performed by using pass/fail criteria for the SuT and generating scenarios with uncertain outcomes close to the pass/fail boundary. The sampling of concrete scenarios will be validated by verifying that their parameter values fall within the parameter space ranges of the corresponding logical scenario. Sanity checks with respect to the handling of missing parameter values or out-of-range parameter values will be performed.

VED: VED will use hand-crafted concrete scenarios or (if time allows) will apply a ready-to-use tool from T3.3 to concretize scenarios using some criteria. Concretization will be validated with simulation execution and verifying that each concrete scenario is successfully executed and leads to slightly different results/KPIs.

Allocate

The initial allocation is always on simulation, going for higher fidelity testing when it is needed. Use case 1.3 will use the environment allocation from the SAF. More in detail, use case 1.3 will use the initial allocation of the test cases to the test instances with the process described in the deliverable D3.3 [5]. All tests that can be performed using simulations will be allocated to virtual testing. However, in a second run, a few cases will be re-allocated to physical or hybrid testing.

ICCS: ICCS will handle the initial allocation of the test cases to the test instances with the process described in the deliverable D3.3 [5].

VED: VED will handle the initial allocation of test cases to test instances using the process outlined in deliverable D3.3 [5].

Execute

Tests will primarily be performed as virtual tests using the simulation pipelines described previously. All data that is needed to evaluate the validity of the “Execute” block, following the criteria outlined in sec 2.2, will be collected automatically.

ICCS: ICCS plans to execute allocated test cases on its V&V virtual simulation framework defined in D4.4 [6], where two pipelines will be implemented, i) virtual testing and ii) hybrid testing with one real agent. Upon completion of sanity checks and results logs completeness checks will be performed to validate execution.

VED: VED will validate the "Execute" block in two phases, adhering to deliverables that define specific validation requirements. The primary objective of this validation is to ensure that test case results meet the input requirements of the “Test Evaluate” and “Coverage” blocks, as outlined in the SAF framework.

Phase 1: VED will initiate validation with simulation testing to identify critical scenarios, ensuring all models, tools, and equipment are fully prepared. Following each test, a **LogCheck** will verify that logs indicate successful test preparation and execution, identifying any setup or execution errors. Test instance reallocation will be performed according to guidelines in Deliverables D3.4 and D3.5. An **RCheck** will confirm that the result data includes all necessary metrics for the "Test Evaluate" block, aligning with the metrics requirements in Deliverable D4.5. Critical scenarios identified in the simulation will then proceed to XiL (Hardware-in-the-Loop) testing in Phase 2.

Phase 2: In this phase, VED will validate the "Execute" block for hybrid testing by preparing a setup that integrates virtual simulation with a real vehicle in the loop. During execution, the validation process will include **LogCheck** and **RCheck** to ensure accurate data collection and metric evaluation. This process will be guided by Deliverable D4.6, ensuring compliance with the requirements of the “Test Evaluate” and “Coverage” blocks in the SAF framework.

Coverage

The coverage analysis on concrete scenario level is based on the ML module developed in T3.3 which aims to automatically generate safety-critical scenarios with optimum space coverage. The optimum coverage is assessed in each iteration until specific criteria are reached (see section 3.3.2.1-ICCS).

ICCS: ICCS will assess coverage on the logical scenario level by using the probabilistic process developed in T3.3. The proposed method allows for i) estimation of the pass/ fail probability of unseen concrete scenarios without the need for execution/simulation, and ii) generation of scenarios close to the pass/fail boundary, which are the hardest to predict their outcomes (high uncertainty). The coverage of the logical scenario’s parameter space will be assessed by estimating the method’s output exhaustively for each scenario parameter instance of the parameter space (which would not be possible if simulation was required). The coverage validation will be performed by drawing scenarios close to the estimated boundary (uncertain), as well as random scenarios in the estimated pass and fail regions and evaluating them in simulation to verify pass or fail decision against estimated by our algorithm pass or fail decision.

VED: VED will perform coverage analysis by using a range of parameter values through virtual testing, following the process developed in T3.3.

Test Evaluate

ICCS & VED: Pass/Fail criteria are based on how many of these tests pass or fail. Pass/fail criteria are also expected by the work in WP2 (ongoing). The pass/fail criteria will be defined based on popular KPIs employed in the perception testing literature and based on experts' feedback. XiL testing will be verified w.r.t offering a different perspective than the virtual one, via recording of new KPIs for safety argumentation. Tests allocated to a hybrid testing environment (XiL) will be validated against their counterparts in simulation.

ICCS: From the individual KPIs results the proximity to the pass/fail boundary will be assessed which will inform on the confidence of the test evaluate output.

Decide

ICCS & VED: ICCS & VED will evaluate if at that stage there is all the needed information from "Coverage" and "Test evaluat" to determine if the SUT is safe or if more testing is required.

3.3.3 Deviations from D7.1

No deviations, but a clear split between ICCS and VED was agreed upon by assigning different selected UCs to each team.

3.4 Use Case 2.1: Traffic jam AD validation – Safety assessment & Decision making

Use Case 2.1 involves testing the SUNRISE SAF with an AD function that controls the ego-vehicle in a traffic jam scenario. The system is expected to control both its longitudinal and lateral position by a combination of braking, and steering commands. All of this is expected to happen while respecting a few constraints, such as a maximum longitudinal speed and a distance to the vehicle in front of the subject vehicle.

This use case involves the following partners:

TNO and the **University of Warwick (UoW)** performed scenario generation for use case 2.1. This generation was based on the key functional scenarios specified in Deliverable D7.1 [1]. These scenarios were identified as key test cases for traffic jam AD. These scenarios were created in using Streetwise (TNO) UoWs Scenario Description Language (SDL) level 2 (UoW) and translated into ASAM OpenSCENARIO. For execution in simulation environments, ASAM

OpenSCENARIO (XOSC) and ASAM OpenDRIVE (XODR) files are generated for these scenarios. The scenarios remain logical, with the ranges specified in the deliverable in D7.1 [1] and D3.2 [3] being used in the ASAM OpenSCENARIO files.

CAF developed the System under Test (SUT) and supported the integration of the System into the virtual testing environment, which implied the addition of certain features to the SUT as well as technical assistance for the integration, to enable AVL to proceed with their validation activities.

AVL and **AVL TR** are responsible for the allocation of the concrete scenarios to the test instances, the creation of the simulation models, the setup of the virtual simulation toolchain, the setup of the automated scenario execution, the integration of the SUT in the closed loop SiL environment and for the implementation and testing of the KPI scripts and test report templates.

BASt is responsible for proving ground and real-world testing regarding Use Case 2.1. A testing method based on the outcomes of Deliverable D4.6 “hybrid and real-world testing and validation approaches” will be demonstrated by the execution of physical test runs including multiple of the 12 defined scenarios (2-A to 2-L in Deliverable D7.1). To show the testability and to validate the test method initially an exemplary state-of-the-art rental vehicle is tested in a black-box approach. This demonstrates the procedure of physical test execution within the Sunrise SAF from the view of a consumer protection entity or the market surveillance of a regulator. It will demonstrate how a vehicle will be equipped with additional sensors to measure the performance of the ADs function while having no further insight in the vehicle's internal technology, e.g. the AD function. The KPI list from D7.1 [1] will be taken into account.

3.4.1 Setup

3.4.1.1 Virtual testing

CAF implemented a series of ROS2 nodes implementing both longitudinal and lateral control intended to be used for virtual testing. These nodes were based on two configurable PID controllers, which respected the above-mentioned constraints. The inputs were the distance to the vehicle in front of the ego-vehicle and a series of “waypoints” relative to the ego-vehicle's position. To ensure smooth control, they are used together with the current speeds of both the ego and the leading vehicle, as well as the current steering angle of the ego-vehicle. In spite of this not directly validating the UC, it enabled AVL to do so, especially with the provided support to integrate said SUT into their simulation framework. Some additional features were added to the controller to ease the integration of the controllers in AVL’s simulation system.

For the virtual testing setup, the model was developed using AVL Model.CONNECT^{TM} simulation software. Model.CONNECT^{TM} serves as the interface to connect multiple software instances. This means, this model incorporates multiple simulation blocks to accurately represent the real-world system. The primary blocks utilized are:

- AVL Vehicle Simulation Model (AVL VSM^{TM})
- Robot Operating System (ROS) communication block

- Esmini

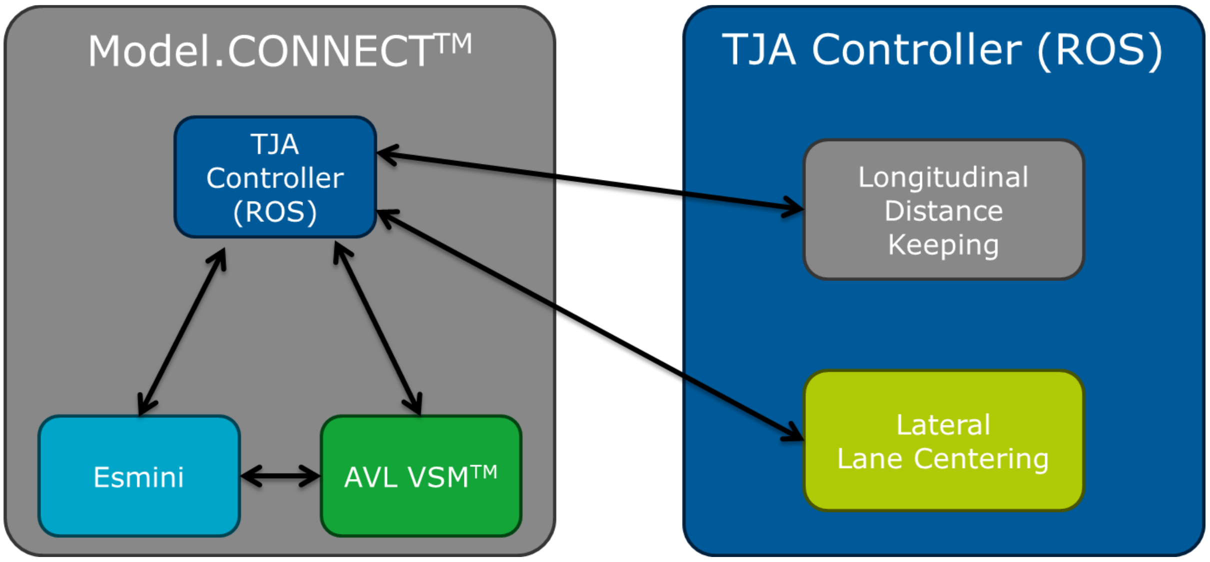


Figure 28: Virtual testing setup

<u>AVL Vehicle Simulation Model (AVL VSM^{TM})</u>

The AVL VSM^{TM} simulation block is responsible for modelling vehicle dynamics and characteristics. It simulates various aspects of the vehicle's behaviour, including acceleration, braking, and steering. The inputs to the VSM block include:

Accelerator pedal position: Determines the vehicle's acceleration.

Brake pedal position: Controls the vehicle's deceleration.

Steering wheel angle: Dictates the vehicle's direction.

These inputs are sourced from the ROS communication block, which runs Python scripts to facilitate communication with the controller.

<u>TJA Controller (ROS)</u>

The ROS communication block serves as the intermediary between the vehicle's sensors and the controller. It is divided into two main functionalities:

Longitudinal Control: This aspect of the controller is responsible for distance keeping. An ideal sensor provides an object list, which includes the distance to the lead vehicle. The controller uses this information to maintain a safe following distance by adjusting the vehicle's speed.

Lateral Control: This aspect of the controller handles lane centering. Using generated waypoints as ground truth, the controller positions the vehicle in the center of the lane. The waypoints provide precise coordinates that the vehicle should follow to stay centered.

<u>Esmini</u>

The Esmini scenario engine is employed for the playback of Open Scenario files. It provides visualization and playback of the ego vehicle and the other included actors from the scenario. The ego vehicle's position, orientation, and velocity are transmitted from the vehicle dynamics block to ensure an accurate representation of the ego vehicle's localization within the virtual environment. This allows a realistic representation of the driving environment and the interactions between the ego vehicle and other road users.

Closed-Loop System

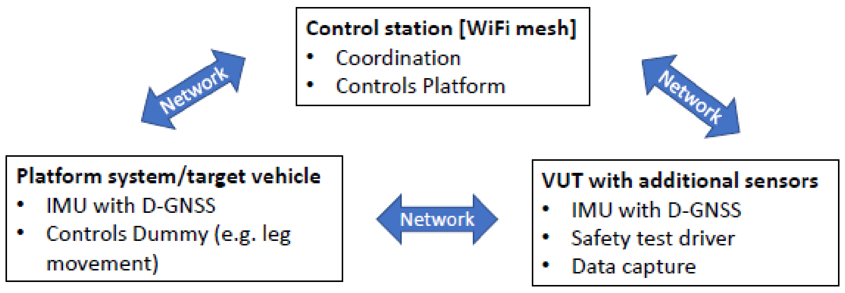
By integrating these three blocks into a closed-loop system, we can simulate the behaviour of the vehicle and the controller across a variety of scenarios defined in the Open Scenario file format. This setup enables a comprehensive virtual testing environment, allowing for the realistic replication of diverse driving situations and conditions. The closed-loop system ensures that the vehicle's responses to different inputs and scenarios are continuously fed back into the simulation, providing a dynamic and interactive testing platform.

This detailed setup allows engineers to test and validate the performance of the vehicle's control systems in a controlled and repeatable manner, ensuring that the vehicle can handle a wide range of driving scenarios safely and effectively.

3.4.1.2 Proving ground and real-world testing

To test safety-critical scenarios in the field of testing ADAS and AD systems multiple test tools can be used nowadays. These tools include targets in the form of vehicles and vulnerable road users (VRUs) which represent the physical and optical properties of these objects (see Figure *30* and Figure *31*). An advantage of these targets is that they are impactable with the vehicle under test (VUT), easy to rebuild after a crash and don't damage the VUT. Self-driving propulsion systems are used to drive the targets and synchronize their position with the VUT. In order to ensure a high level of position accuracy, inertial measurement systems (IMUs) are used, which are installed in all objects involved in the test (VUT and targets). In combination with a Differential Global Navigation Satellite System (D-GNSS), accuracies in the range of a few centimetres can be achieved. A network including a WiFi mesh is built up to ensure a constant data flow between all objects included in a test (see Figure *29*).

For the test execution, reference is made below to the test procedures in UN Regulation No. 157 (R157 - Automated Lane Keeping System) from 2021, on which Use Case 2.1 (TJA) with the restriction to maximum vehicle speeds of up to 60 km/h is based. In the amended version of R157 from 2022, the maximum vehicle speed during an activated ALKS manoeuvre was



increased to 130 km/h.

Figure 29: Network and Connection setup for Test Track testing with a VUT, a dummy or target vehicle and a control station

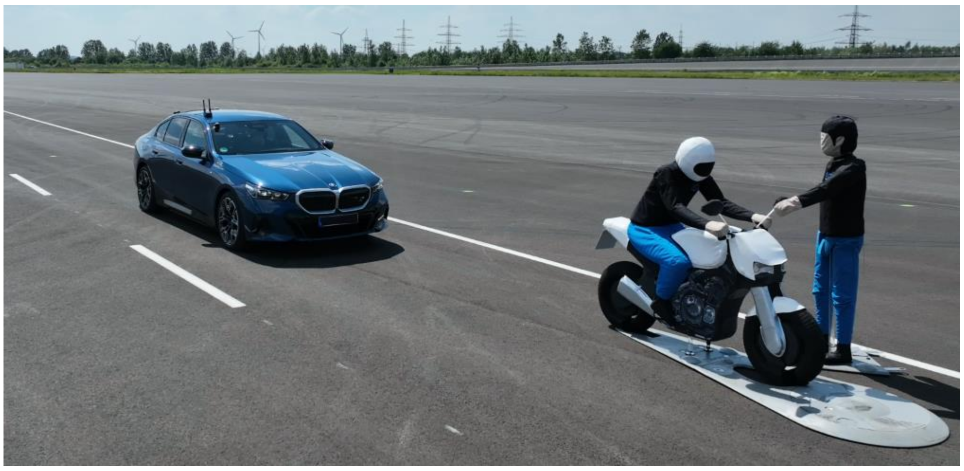


Figure 30: Proving ground testing setup with VUT, Motorcycle and Pedestrian Dummy in a blocked lane scenario

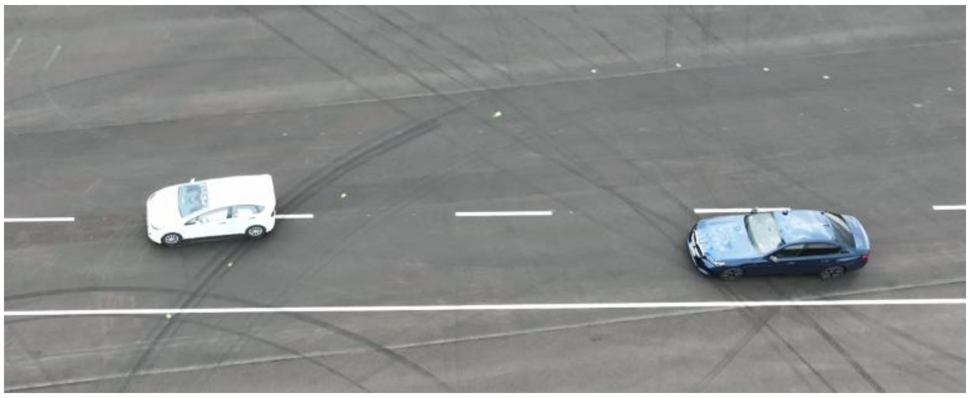


Figure 31: Proving ground testing setup with VUT and Car Target in a cut-in scenario

As a test environment, a proving ground is chosen to evaluate the safety critical test scenarios in a safe and isolated area. To ensure safety during the test execution and to have enough space to stage the scenarios a vehicle dynamic area is chosen as an appropriate area on the test track. To provide a realistic scenario within the ODD of the TJA/ALKS system lane markings are required on the proving ground. If a test organisation or regulator will test these systems independent from the OEM (e.g. consumer protection or market surveillance of a regulator) they can’t test the full functionality of these systems on a test track due to the limitation called “Geofencing”. This means that the ODD of most of the systems from SAE L3 on or higher (including SAE L2 with hands-off functionality) is limited to specific road types (e.g. highways) and the system can’t be activated in restriction areas, such as construction sites.

To validate the performance seen on the test track and test the full functionality of the system additional tests on real roads can be carried out. To ensure safety during these real road drives can only be executed in a safe uncritically range of test parameters. For this, a test setup with two cars (VUT and a target vehicle) will be evaluated in the Sunrise project (see Figure *32*). The VUT (ego) and the second vehicle (target) will be equipped with measurement equipment that records their individual position data and in addition the relative data between these two vehicles.

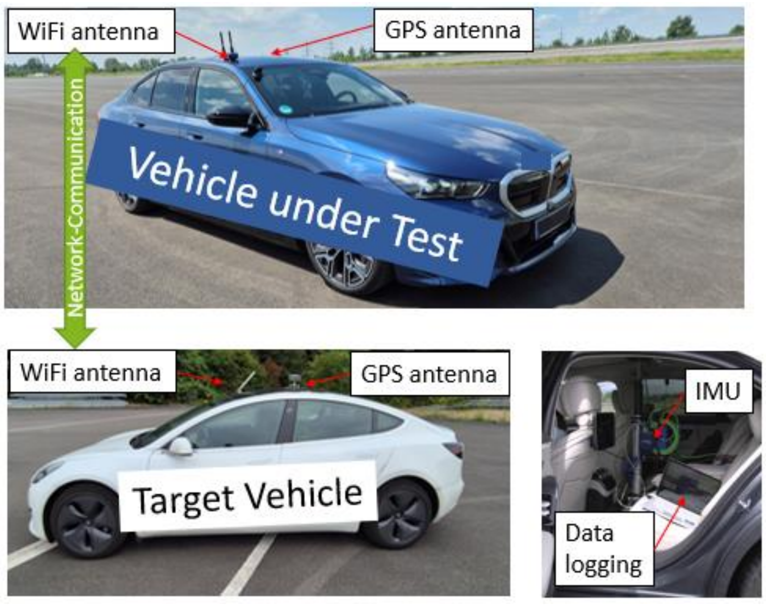


Figure 32: Test setup for real road drives with two real cars and network connection for relative data calculation

In terms of SUNRISE UC 2.1 the two-step approach taken from the EU regulation No. 157 (ALKS) will be followed: First safety backup tests on a test track and after that testing on real roads.

Outcomes of the physical testing will be the demonstration of the toolchain of test tools (e.g. targets) and measurement devices for the execution of proving ground and real-world testing of the test scenarios of UC 2.1. For the test evaluation, the relevant KPIs and metrics defined in D7.1 [1] are calculated and evaluated. External requirements from the EU regulation are taken into account in order to demonstrate the “Decide” block from the SAF.

3.4.2 Covered aspects of the SAF

The table below shows the contributions to the SAF block validation per partner for UC2.1.

Table 6: Contributions to the SAF validation per Partner in UC2.1

Partner/ Block	AVL	BASt	CAF	TNO	UoW
SUNRISE DF				x	x
Query & Concretise	x				
Allocate	x				
Execute	x	x	x		
Test Evaluate	x	x			
Coverage					
Decide		x			

SUNRISE DF

The function of the federated layer in this context is to allow for queried scenarios to be selected for testing, which have been requested and verified as provided in the correct format. The federated layer is not developed at this stage. The scenarios provided by TNO and UoW mimic the use of the federated layer in that the requirements for the scenarios were established in the SUNRISE deliverable D7.1 [1] and queried to meet said requirements. The format of the scenarios from both TNO and UoW, is consistent as OpenSCENARIO 1.0 and OpenDRIVE files. Due to the state of readiness of the Federated DF, the scenarios were created outside but will be validated by a series of queries to the DF and subsequent analysis to ensure they align.

TNO: TNO hosts an online SCDB (Streetwise) which will be queried based on the requirements of the use case set out by T7.1 for UC 2.1 by the federated layer.

UoW: UoW hosts a SCDB (Safety Pool^{TM}) which will be queried based on the requirements of the use case set out by T7.1 for UC 2.1 by the federated layer.

Query & Concretise

A selection of logical scenarios including parameter ranges were created in deliverable D7.1 [1]. Those scenarios will be processed to provide concrete scenarios by query & concretise block, and already prepared concrete scenarios from Safety Pool will be quired and forwarded to allocation block.

AVL: AVL and AVL TR will contribute to the validation of the Query & Concretise block by creating concrete scenarios from logical scenarios by using the methodology described in deliverable D3.4.

Allocate

Initial allocation will be done by using the process provided by deliverable D3.3 [5]. To validate this allocation decision-making process, a specific scenario from UC 2.1 is chosen and compared with different test capabilities on virtual environments and proving ground tests. Also, the allocation for the scenarios will be done, based on the needed fidelity level. This means, that for low-fidelity testing, using the virtual simulation approach will be sufficient and for high-fidelity testing proving ground tests will be needed.

AVL: AVL and AVL TR will allocate concrete scenarios to test instances as described in D3.3 [5] to validate defined scenarios in D7.1 [1]. Selected scenarios from the simulation, which are validated, then will be executed in the proving ground.

Execute

Tests will be performed as virtual tests and physical tests. For virtual testing efficient orchestration requires test scenarios to be machine readable, i.e., by using OpenSCENARIO format. All data that is needed to evaluate the validity and results of tests will be collected automatically. For physical testing a concrete test scenario definition is needed that provides relevant information to synchronize the state-of-the-art test tools for ADAS- and AD-testing (e.g. targets with propulsion systems) with the VUT. For example, tolerances of measurement devices and technical limits of propulsion systems for targets have to be taken into account. It will be demonstrated what measurement devices are needed to collect all relevant data channels for further analysis in the section “Test Evaluate” with regard to the defined KPIs/metrics in D7.1 [1]. The generated test results must fit the output of the block called “ODD & Behaviour, External Requirements, Test Objectives”.

AVL: AVL and AVL TR will demonstrate automated scenario executions in the simulation toolchain. Additionally, results will be used for the KPI calculations and test reports.

BASt: BASt will demonstrate the execution of physical test runs in a black-box approach (based on the outcomes of T4.6), like it will be done by consumer testing or market surveillance of a regulator using the Sunrise SAF.

CAF: CAF will develop the System under Test (SUT) and support the integration of the System into the virtual testing environment. This doesn’t validate the UC, but rather enables AVL’s activities.

Test Evaluate

Based on the outcomes of the test execution the test results for virtual and physical testing will be evaluated regarding the validation metrics from the upcoming deliverable D3.5 and KPIs defined in D7.1 [1]. Based on the evaluation results which will be returned to the SAF, test cases will be re-allocated or scenarios with updated parameters by using methods in D3.3 [5] will be tested.

AVL: AVL will use KPIs, defined in D7.1 [1] and D3.2 [3], KPI calculation scripts and test report templates will be prepared for both simulation and proving ground test results evaluations.

BASt: BASt will evaluate the test data after test execution by taking into account the metrics/KPIs defined in D7.1 [1] and T3.5.

Decide

To evaluate the performance of the system in the TJA Use Case, the SUNRISE Task 3.5 will provide pass/fail criteria. These assessment criteria are KPIs, metrics and other requirements. In the EU regulation R157, on which UC 2.1 is based, the requirements are openly formulated (e.g. system must stick to the traffic rules and avoid collisions up to a specific TTC). The SAF will specify concrete evaluation metrics related to the defined test scenarios and based on the general requirements from the regulations.

BASt: BASt will validate the "Decide"-block after the evaluation of the test results by taking into account external requirements from the EU R157, on which UC2.1 is based.

3.5 Use case 3.1: Highway AD validation – Map based perception & decision making

Use Case 3.1 involves testing the SUNRISE SAF with map-based AD functions for highway scenarios. The systems under test will include an advanced ACC/HWP designed to control longitudinal dynamics based on HD map data and onboard sensors.

There are two defined functional scenarios: adapting speed to varying speed limits and varying road curvature, using information from sensors and maps. A third functional scenario, green driving on slopes, initially proposed in SUNRISE deliverable D7.1 [1], has been dismissed.

This Use Case involves the following partners:

UNITN will conduct simulations in the IPG CarMaker environment using a model of the systems under test. Map data will be provided to the self-driving agent by the host CarMaker environment, and speed limit signs can also be encoded.

IDI will contribute by validating the systems by conducting comprehensive tests at a proving ground. This involves not only designing and integrating all the autonomous system software and hardware but also incorporating the specified use cases. The integration process will ensure that all components work together, meeting the requirements for the use case. Once the systems are fully integrated, IDI will execute these test cases in a controlled environment at the proving ground.

IDI DE will conduct simulations in the MATHWORKS Automated Driving Toolbox environment. Map data will be extracted from the ASAM OpenDRIVE road network files and provided as ground-truth information to the map-based HWP implemented as MATHWORKS Simulink model. The University of Warwick performed scenario generation for the use case. This generation was based on the four functional scenarios specified in Deliverable D7.1 [1], defining the use-case. These scenarios were created in WMGs SDL level 2 and translated into ASAM OpenSCENARIO for execution in simulation environments, XOSC and XODR files are generated for these scenarios.

ika is responsible for implementing the parameter sampling strategies developed within the SUNRISE project to facilitate testing in this use case. This includes both the generation of an initial parameter set and the development of a methodology for optimizing samples within the parameter space, taking into account the scenarios generated by UoW. Additionally, ika supports IDI in integrating sensor models into the simulation toolchain.

3.5.1 Setup

3.5.1.1 Virtual testing

Virtual testing is carried out by UNITN and IDI DE.

UNITN will use the IPG CarMaker environment described in deliverable D4.3 [9], section 2 (2.1.4 for the test case manager, 2.2.4 for the environment, 2.3.5 for the test vehicle, 2.4.2 for the traffic agents).

IDI DE will use ASAM OpenSCENARIO files containing the scenarios linked to the road network files implemented in ASAM OpenDRIVE format containing the map information. Esmini will be used as a scenario player to provide the trajectories to the MATLAB Automated Driving Toolbox simulation. The map data will be extracted from the OpenDRIVE files and provided to the SUT. The scenarios will be parameterised via Python scripts doing the sampling and exported into OpenDRIVE/OpenSCENARIO format by making use of the Python library scenario generation.

3.5.1.2 Proving ground testing

IDI will conduct tests at the proving grounds located in specialized facilities at IDIADA Headquarters in Spain. For safety and testing purposes, this will be an isolated area, equipped with all the necessary elements to conduct the tests, such as traffic signs and other vehicles. These tests will focus on executing the scenarios described in the deliverable D7.1 [1], including adapting speed according to the new speed limit and modifying velocity in response to road curvature.

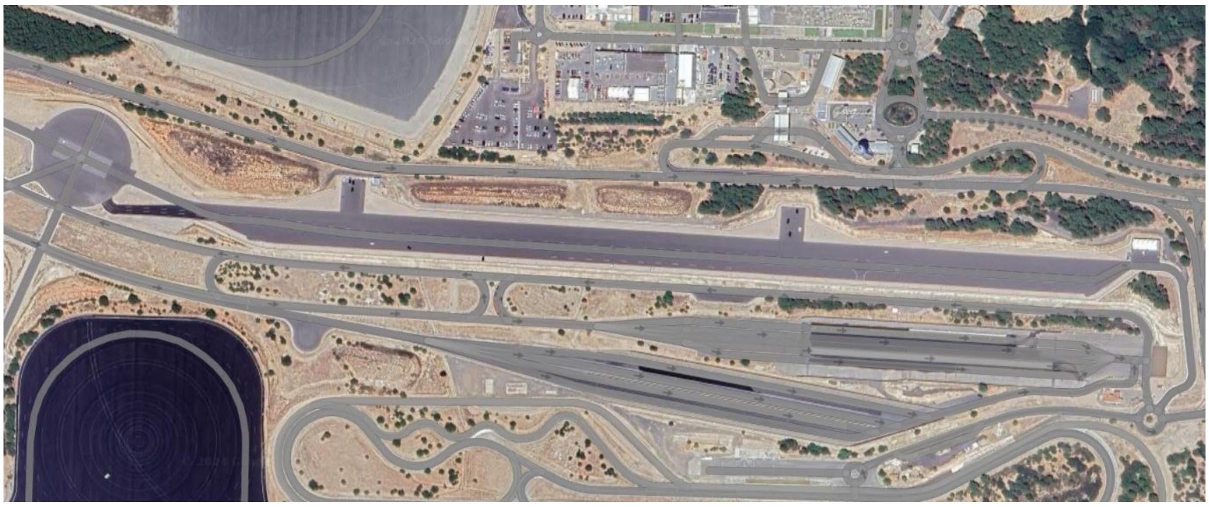


Figure 33: Specialized ADAS Proving Ground used for Use Case 3.1 and 3.2 (IDI)

The primary objective of these tests is to correlate simulations with real-world data and to parameterize the results for detailed analysis, leading to clear conclusions. Additionally, the tests will be designed to demonstrate that the vehicle's safety is enhanced when it receives and understands information from maps, including the characteristics of events depicted on those maps.

Furthermore, IDI will employ internal control and perception algorithms to maintain complete control over the tests, ensuring the reliability of conclusions and the extraction of all necessary data.

The autonomous driving prototype used by IDI will be equipped with a perception system that includes cameras and RADARs. Additionally, the vehicle will feature a comprehensive autonomous driving pipeline, incorporating sensor fusion, planning, localization, and control algorithms, all implemented using MATHWORKS tools and deployed in rapid control prototyping devices such as Speedgoat. It will also be outfitted with IMUs, GPS, and other sensors to collect the necessary data for the test cases.

For safety reasons, an authorized driver will be present in all vehicles participating in the test cases, ready to stop the testing at any moment if required.

3.5.2 Covered aspects of the SAF

The ODD and the two functional scenarios are defined in SUNRISE deliverable D7.1 [1]. The table below shows the contributions to the SAF block validation per partner for UC3.1.

Table 7: Contributions to the SAF validation per Partner in UC3.1

Partner/ Block	IDI	ika	UNITN	UoW
SUNRISE DF				x

Query & Concretise		x	x	
Allocate	x	x	x	
Execute	x	x	x	
Test Evaluate	x		x	
Coverage	x	x*	x	
Decide	x		x	

*only parameter space coverage

SUNRISE DF

UoW (WMG) will provide scenarios tailored to the requirements of the UC and provide them in the required common format for execution, except for V2X elements which are not supported directly in ASAM OpenSCENARIO. Despite the scenarios being created before the deployment of the Federated DF, UoW will validate the scenarios as though they were output from the Federated DF. UoW will test several queries based on different ODDs and analyse whether the logical scenarios used in this use case (and their parameter ranges) match the ODDs and the expectations of the experts (know the content of the SCDB as host).

Query & Concretise

Queries will be created to search for scenarios via the federated SCDB. The suggested scenarios will be compared to the ones defined by the experts in WP7.

Most of the scenarios and ranges are coming from UoW, the bulk of the scenarios come from translations of the abstract scenarios defined as 'key' in the D7.1 [1] report. Whilst the provision of scenarios from an existing database such as Safety Pool^{TM} would provide real measurement data, the novelty of the map data described in the use case prevented this. The scenario subspace generation methodology is in progress and will be provided by ika. The UC will test the completeness by checking whether the concrete scenarios missed any critical case (this is done by sampling the parameter space with many new randomly generated samples to verify that nothing was missed). The efficiency of the concrete scenarios will be tested by UNITN with a ML surrogate and bootstrapping aggregation method (this is done by comparing the number of concrete scenarios to the minimum number that still captures all the important features). Additionally, the obtained behaviours of participants and the quantities that must be measured can be checked for completeness.

ika: ika generates concrete scenarios for simulation based on the methodology developed in WP3. The case generation process requires input from the query process, which will be validated against the criteria outlined in Chapter 2 for the logical scenarios defined in the use case. Additionally, the sampling methodology will be evaluated by assessing how well the parameter space is covered by the KPIs specified in the use case.

UNITN: UNITN will compare logical scenarios from the use case to internally defined scenarios.

Allocate

The UC will use the test instance allocation from the SAF following the allocation process from Deliverable D3.3 [5]. This step will show whether each test run can be allocated to the virtual simulation test instances. In the second step, a few cases will be reallocated to physical testing following the initial allocation process or after execution and test case validation of the re-allocation process from Deliverable D3.5.

IDI: IDI will apply the initial allocation process in chapters 3 and 4 of Deliverable D3.3 [5] and validate the test cases according to the upcoming Deliverable D3.5. Virtual simulation and proving ground testing will be used as test instances to validate the allocation. It will be verified that test cases can be allocated to test instances and checked whether the initial allocation indicates test instances of unexpected high fidelity. Potential reallocation will be applied according to Deliverable D3.5.

ika: ika supports the allocation validation of our use case partners by providing the connection to the development in WP3.

UNITN: UNITN conducts the cases that are assigned to simulations. By evaluating whether the simulations successfully generated accurate evaluation metrics, one can determine whether the simulation test instances were appropriate or if the allocation block should have indicated test instances with higher fidelity.

Execute

The SAF will provide the test objectives that will be compared to expert definitions in WP7 (KPI, metrics and purpose of testing), behaviours (e.g., what might be the misbehaviour of violating pedestrian and car) or other requirements. Execution will quantify the provided metrics. After executing the concrete test scenarios on the allocated test instance, it will be checked whether the metrics needed in the following blocks Test Evaluate and Decide can be computed from the obtained log and result data. In case of missing data (e.g., a failed simulation), the block which causes the problem will be identified.

IDI: IDI-DE will run the simulations using the framework described in section 3.5.1.1 and IDI will conduct proving ground testing as described in section 3.5.1.2. It will be reported whether each concrete scenario was executed correctly and the information needed in the following blocks, Test Evaluate and Decide, is available. In case of missing data (e.g., a failed simulation), the block which causes the problem will be identified.

ika: ika supports partners in the validation of the sensor models. For the validation, the sensor information from real-world tests is compared to the simulated sensor output.

UNITN: UNITN runs the simulations and reports whether each concrete scenario was correctly executed or if it was not completed and why.

Coverage

The UC will test the coverage of the parameter space by sampling the parameter space with many randomly generated samples in addition to those provided by the SAF. An alternative approach consists of creating an approximant function with known uncertainty (via bootstrapping) and using it for comparison with the coverage analysis made in SUNRISE workflow. Coverage will be repeated after successive iterations that may be created by the SAF.

IDI: IDI and IDI-DE will provide their results from virtual simulations and proving ground testing to ika and UNITN for validation of the coverage.

Ika: evaluates the coverage of cases within the parameter space required for the methodology. By comparing the results against simulations and real-world testing, we validate that the surrogate model, which generates the test cases, effectively approximates the underlying distributions of the KPIs across the parameter space. However, this validation only addresses part of the coverage block, as scenario coverage cannot be assessed.

UNITN: UNITN cross checks SAF coverage from ika with the bootstrapping aggregation methods mentioned above, in query and concretize.

Test Evaluate

The tests from virtual simulations and on the proving ground will be evaluated for the safety performance of the SUT. This evaluation will be based on the metrics provided by the SAF's "Query & Concretise" block and checked for sufficient information for a safety assessment. Finally, the test evaluation will be compared to the expert's judgement.

IDI: For each concrete scenario executed in virtual simulation or on proving ground, the metrics from the SAF for evaluating the safety performance of the HWP and the validity of the test runs will be returned. In case a metric cannot be calculated, the block which causes the problem will be identified. Based on experience, it will be checked whether the metrics from the SAF are appropriate for measuring the safety performance of the HWP and the validity of the test run.

UNITN: UNITN returns the performance indicators per concrete scenario and compares the SAF evaluations by ika with internal evaluations.

Decide

The guidelines in upcoming Deliverable D2.3 for a final pass/fail safety evaluation (or pass conditioned to restricted subsets of the parameter space) will be followed. These guidelines and the result will be checked for plausibility and feasibility.

UNITN: UNITN will seek to falsify the SAF decision with one of the following methods: a) new independent test samples (either in the parameter space or with new parameters that were not considered in the logical scenario), b) with a surrogate ML model (bootstrapping aggregation method) of the test results across the logical scenario (as in UNITN's paper under review)

IDI: IDI will follow the guidelines in the upcoming Deliverable D2.3 to derive a safety evaluation of the HWP from the SAF. Based on experience, these guidelines and the result will be checked for plausibility and feasibility.

3.5.3 Deviations from D7.1

After CRF withdrawal the same SUT for virtual simulations and physical testing holds for the system of IDI. In the other cases, the physical and simulated systems are different. However, since they must be acceptable for humans, they must be compatible and could be loosely compared against naturalistic driving criteria. Additionally, as mentioned in the introduction before the third scenario for green speed adaptation to slight road slopes has been dismissed for this Use Case 3.1.

3.6 Use case 3.2: Highway AD validation – Cooperative perception & decision making & control

The main aim of sub-UC 3.2 will be to demonstrate how safety could be improved on motorways by including cooperative functions in the HWP system. One example of this HWP functionality is the leveraging and upgrading of the driver assistance functionality developed previously in C-ACC from sub-UC 1.2.

The main functionality of an HWP or ALKS responsible for longitudinal control and therefore for safety is the ACC system which automatically adjusts the vehicle speed to maintain a safe distance from a vehicle ahead. V2V enables the extension to C-ACC concept thanks to additional information obtained from connected vehicles ahead. The system shall be capable of communicating with other road users using V2X messages (e.g., CAM or DENM) for cooperative perception and sharing its perception or map information with others as well.

C-ACC is a driver assistance system implemented on top of ACC, it may be combined with other functionalities such as FCW (Forward Collision Warning), EEBL (Emergency Electronic Brake Light) for increased safety. It receives information from different sources, onboard and through connection, about objects surrounding the vehicle and road features.

Today there are projects like MuCCA where cooperative manoeuvres using V2V communication have been demonstrated on proving grounds by the usage of a few scenarios defined by experts. Furthermore, there are projects like the EU-funded research project 5G-CARMEN (5G for Connected and Automated Road Mobility in the European UnioN) [10], where cooperative manoeuvres using V2X communication have been demonstrated in specific cross-border scenarios on public roads along the highway corridor Munich-Bologna with the goal, defined by experts, to test the continuity of 5G services for CCAMs and to measure 5G KPIs for automated driving.

As can be seen, the demonstration of the effectiveness of V2X communication in cooperative manoeuvres has been limited to specific scenarios defined by experts and mostly related to the operability analysis of connectivity services. Therefore, the goal is to go beyond the defined limitations regarding the scenarios and the connectivity studies. In fact, by applying the SUNRISE SAF, the ODD coverage can be demonstrated, not spatially or functionally to specific operating fields and conditions. This is possible by taking advantage of different virtual validations, e.g., usage of extended realistic scenarios where cooperative manoeuvres between agents can be proven by variating all the interesting parameters, even in corner cases or in safety-critical conditions. After extensive virtual tests demonstrating the virtual ODD coverage, the designed cooperative functions can be proven in real-world scenarios regarding the main identified use cases and parameters, in order to provide a sufficient test coverage of the ODD.

3.6.1 Setup

Use Case 3.2 involves testing the SUNRISE SAF with cooperative AD functions in highway scenarios. The systems being tested will include an advanced Adaptive Cruise Control (ACC) with Vehicle-to-Everything (V2X) communication, such as Cooperative Awareness Message (CAM) data from other vehicles, which is expected to control the longitudinal dynamics.

There are four defined functional scenarios:

a) ACC adapting speed to hidden leading vehicles,

b) ACC adapting to hidden decelerating vehicles,

c) ACC anticipating cut-in cooperative vehicles, and

d) ACC reacting to the loss of control (ESC/ABS) of a leading vehicle.

This Use Case involves the following partners:

UNITN will conduct simulations in the IPG CarMaker environment using a model of the systems under test. V2V communication will be simulated via SimNet, which is a CarMaker add-on that allows for multi-agent simulations.

IDI will contribute by validating the systems and conducting comprehensive tests at a proving ground. This involves not only designing and integrating all the autonomous system software and hardware but also incorporating the specified use cases. The integration process will ensure that all components work together, meeting the requirements for the use cases. Once the systems are fully integrated, IDI will execute these use cases in a controlled environment at the proving ground.

IDI DE will conduct simulations in the MATHWORKS Automated Driving Toolbox environment. For the V2V communication, the models of the Automated Driving Toolbox will be adapted in order to exchange the same messages as real prototype vehicles following the ETSI standards for communication. The University of Warwick performed scenario generation for the use case. This generation was based on the four defined functional scenarios above. These scenarios were created in UoWs SDL level 2 due to its capability to describe V2X communications in scenarios. These messages and capabilities are not currently translatable into ASAM OpenSCENARIO for execution in simulation environments, XOSC and XODR files are generated for these scenarios using a conversion toolchain that ignores the V2X elements that are not currently translatable.

The activities of ika in this use case mirror the on in use case 3.1. ika is responsible for implementing the parameter sampling strategies developed within the SUNRISE project to facilitate testing in this use case. This includes both the generation of an initial parameter set and the development of a methodology for optimizing samples within the parameter space, taking into account the scenarios generated by UoW. Additionally, ika supports IDI in integrating sensor models into the simulation toolchain.

3.6.1.1 Virtual testing

Virtual testing is carried out by UNITN and IDI / IDI DE.

UNITN will use the IPG CarMaker environment described in deliverable D4.3 [9], section 2 (2.1.4 for test case manager, 2.2.4 for the environment, 2.3.5 for the test vehicle, 2.4.2 for the traffic agents, 2.5.1 for connectivity).

IDI DE will use ASAM OpenSCENARIO files containing the scenarios linked to the road network files implemented in ASAM OpenDRIVE format. The OpenSCENARIO files will also contain the V2X messaging information (e.g. message type, content, times) by making use of “UserDefinedActions”. This incorporation will be manually done for the scenarios provided by UoW based on the information provided in the SDL files or the whole SDL scenarios from UoW will be reimplemented in scenariogeneration python module to generate XOSC. Esmini will be used as a scenario player to provide the trajectories to the MATLAB Automated Driving Toolbox simulation. An additional Python script will be used to extract the V2X information from OpenSCENARIO and will be provided to the MATHWORKS simulation, too. The connectivity part will be implemented in the MATHWORKS Simulink model making use of models for sender and receiver as part of the Automated Driving Toolbox which will be adapted according to the Connectivity strategy that will be used for the prototype vehicle. The scenarios will be sampled and parameterised via Python scripts and exported into OpenSCENARIO/OpenDRIVE by making use of the Python library scenariogeneration.

3.6.1.2 Proving ground testing

IDI will conduct tests at the proving grounds located in specialized facilities at the IDIADA Headquarters in Spain. For safety and testing purposes, this will be an isolated area, equipped with all the necessary elements such as traffic signs and other vehicles to facilitate the tests. These tests will focus on executing the scenarios described in SUNRISE deliverable D7.1 [1], including cooperative ACC, deceleration of the vehicle in front, cut-in into the ego vehicle's lane, and vehicle control loss.

The primary objective of these tests is to correlate simulations with real-world data and to parameterize the results for detailed analysis, leading to clear conclusions. Additionally, the tests will be designed to demonstrate that overall road safety is enhanced when vehicles share information, communicate with each other, and utilize cooperative functions.

Furthermore, IDI will employ internal communication, control, and perception algorithms to maintain complete control over the tests, ensuring the reliability of conclusions and the extraction of all necessary data.

The autonomous driving prototype used by IDI will be equipped with a perception system that includes cameras and RADARs. Additionally, the vehicle will feature a comprehensive autonomous driving pipeline, incorporating sensor fusion, planning, localization, and control algorithms, all implemented using MATHWORKS tools and deployed in a rapid control prototyping device such as Speedgoat. It will also be outfitted with IMUs, GPS, and other sensors to collect the necessary data for the test cases.

The prototypes that do not manoeuvre using autonomous algorithms but participate in the use cases will be equipped to communicate with the autonomous prototype, although they will be fully driven by a driver.

For safety reasons, an authorized driver will be present in all vehicles participating in the use cases, ready to stop the testing at any moment if required.

3.6.2 Covered aspects of the SAF

The ODD and the functional scenarios are defined in SUNRISE deliverable D7.1 [1]. The table below shows the contributions to the SAF block validation per partner for UC3.2.

Table 8: Contributions to the SAF validation per Partner in UC3.2

Partner/ Block	IDI	ika	UNITN	UoW
SUNRISE DF				x
Query & Concretise		x	x	
Allocate	x	x	x	
Execute	x	x	x	-
Test Evaluate	x		x	-

Coverage	x	x*	x	-
Decide	x	-	x	-

*only parameter space coverage

SUNRISE DF

UoW (WMG) will provide scenarios tailored to the requirements of the UC and provide them in the required common format for execution, with the exception of V2X elements which are not supported directly in ASAM OpenSCENARIO. As for use case 3.1 above the UoW created the scenarios for use in testing when the federated DF was not available for use. UoW will validate the Federated DF by testing several queries based on different ODDs and analysing if the logical scenarios provided for the use case as an outcome (and their parameter ranges) match the ODDs and the expectations of the experts (know the content of the SCDB as host) for the scenarios provided.

Query & Concretise

Queries will be created to search for scenarios via the federated SCDB. The suggested scenarios will be compared to the ones defined by the experts in WP7.

Most of the scenarios and ranges are coming from UoW, the bulk of the scenarios come from translations of the abstract scenarios defined as ‘key’ in the D7.1 [1] report. Whilst the provision of scenarios from an existing database such as Safety Pool^{TM} would provide real measurement data, the novelty of the communications described in the use case prevented this. The scenario subspace methodology is in progress and will be provided by ika. The UC will test the completeness by checking whether the concrete scenarios missed any critical case (this is done by sampling the parameter space with many new randomly generated samples to verify that nothing was missed). The efficiency of the concrete scenarios will be tested by UNITN with a ML surrogate model and bootstrapping aggregation method (this is done by comparing the number of concrete scenarios to the minimum number that still captures all the important features). Additionally, the obtained behaviours of participants and the quantities that must be measured can be checked for completeness.

ika: ika generates concrete scenarios for simulation based on the methodology developed in WP3. The case generation process requires input from the query process, which will be validated against the criteria outlined in Chapter 2 for the logical scenarios defined in the use case. Additionally, the sampling methodology will be evaluated by assessing how well the parameter space is covered by the KPIs specified in the use case.

UNITN: UNITN will compare logical scenarios from the use case to internally defined scenarios.

Allocate

The UC will use the test instance allocation from the SAF following the allocation process from Deliverable D3.3 [5]. This step will show whether each test run can be allocated to the virtual simulation test instances. In the second step, a few cases will be reallocated to physical testing following the initial allocation process or after execution and test case validation of the re-allocation process from Deliverable D3.5.

ika: ika supports the allocation validation of our use case partners by providing the connection to the development in WP3.

UNITN: UNITN conducts the cases that are assigned to simulations. By evaluating whether the simulations successfully generated accurate evaluation metrics, one can determine whether the simulation test instances were appropriate or if the allocation block should have indicated test instances with higher fidelity.

IDI: IDI will apply the initial allocation process in chapters 3 and 4 of Deliverable D3.3 [5] and validate the test cases according to the upcoming Deliverable D3.5. Virtual simulation and proving ground testing will be used as test instances to validate the allocation. It will be verified that test cases can be allocated to test instances and checked whether the initial allocation indicates test instances of unexpected high fidelity. Potential reallocation will be applied according to Deliverable D3.5.

Execute

The SAF will provide the test objectives that will be compared to expert definitions in WP7 (KPI, metrics and purpose of testing), behaviours (e.g., what might be the misbehaviour of violating pedestrian and car) and any other requirements. Execution will quantify the provided metrics. After executing the concrete test scenarios on the allocated test instance, it will be checked whether the metrics needed in the following blocks Test Evaluate and Decide can be computed from the obtained log and result data. In case of missing data (e.g., a failed simulation), the block which causes the problem will be identified.

IDI: IDI-DE will run the simulations using the framework described in section 3.6.1.1 and IDI will conduct proving ground testing as described in section 3.6.1.2. It will be reported whether each concrete scenario was executed correctly and the information needed in the following blocks, Test Evaluate and Decide, is available. In case of missing data (e.g., a failed simulation), the block which causes the problem will be identified.

ika: ika supports partners in the validation of the sensor models. For the validation, the sensor information from real-world tests is compared to the simulated sensor output.

UNITN: UNITN runs the simulations and reports whether each concrete scenario was correctly executed or if it was not completed and why.

Coverage

The UC will test the coverage of the parameter space by sampling the parameter space with many randomly generated samples in addition to those provided by the SAF. An alternative approach consists of creating an approximant function with known uncertainty (via bootstrapping) and using it for comparison with the coverage analysis made in SUNRISE workflow. Coverage will be repeated after successive iterations that may be created by the SAF.

IDI: IDI and IDI-DE will provide their results from virtual simulations and proving ground testing to ika and UNITN for validation of the coverage.

ika: evaluates the coverage of cases within the parameter space required for the methodology. By comparing the results against simulations and real-world testing, we validate that the surrogate model, which generates the test cases, effectively approximates the underlying distributions of the KPIs across the parameter space. However, this validation only addresses part of the coverage block, as scenario coverage cannot be assessed.

UNITN: Compare SAF coverage with the ML bootstrapping aggregation method mentioned in UC3.1 (section 3.5.2).

<u>Test Evaluate</u>

The tests from virtual simulations and on the proving ground will be evaluated for the safety performance of the SUT. This evaluation will be based on the metrics provided by the SAF's "Query & Concretise" block and checked for sufficient information for a safety assessment. Finally, the test evaluation will be compared to the expert's judgement.

UNITN: UNITN returns the performance indicators per concrete scenario and compares ika evaluations based on SAF with internal evaluations.

IDI: For each concrete scenario executed in virtual simulation or on proving ground, the metrics from the SAF for evaluating the safety performance of the HWP and the validity of the test runs will be returned. In case a metric cannot be calculated, the block which causes the problem will be identified. Based on experience, it will be checked whether the metrics from the SAF are appropriate for measuring the safety performance of the HWP and the validity of the test run.

<u>Decide</u>

The guidelines in upcoming Deliverable D2.3 for a final pass/fail safety evaluation (or pass conditioned to restricted subsets of the parameter space) will be followed. These guidelines and the result will be checked for plausibility and feasibility.

UNITN: UNITN will seek to falsify the SAF decision with one of the following methods: a) new independent test samples (either in the parameter space or with new parameters that were not considered in the logical scenario), b) with a surrogate ML model (bootstrapping aggregation method) of the test results across the logical scenario (as in UNITN’s paper under review)

IDI: IDI will follow the guidelines in the upcoming Deliverable D2.3 to derive a safety evaluation of the HWP from the SAF. Based on experience, these guidelines and the result will be checked for plausibility and feasibility.

3.6.3 Deviations from D7.1

After CRF withdrawal the same SUT for virtual simulations and physical testing holds for the system of IDI. Additionally, IDI will not perform the fourth scenario for the Use Case 3.2 mentioned in D7.1 [1] which is the “loss of control” scenario. In the other cases, the physical and simulated systems are different. However, since they must be acceptable for humans, they must be compatible and could be loosely compared against naturalistic driving criteria.

3.7 Use case 4.1: Freight vehicle automated parking validation – Truck low-speed perception & decision making

Confined areas, with perimeter protections and a reduced risk of unauthorised entry, offer the advantage of a well-defined ODD. These environments and highway use cases are expected to be among the first to support deploying highly automated (L4) heavy vehicle functions, aspects that make them essential to SAF validation efforts.

The reverse parking of a truck with a semitrailer is a strong use case for validating specific aspects of the SAF due to the controlled environment and limited range of possible scenarios. Validation in such spaces benefits from lower speeds and regulated traffic conditions, although the presence of mixed traffic and humans introduces additional complexity [11].

Efficient validation methods and toolchains are necessary to ensure the vehicle’s performance meets safety and operational standards. Infrastructure support, such as high-bandwidth, low-latency connectivity for remote monitoring and traffic control, and precise digital mapping, are key enablers of the technology.

For validation, demonstration activities are essential to showcase the utility and safety of the automated system in real-world logistics scenarios. The SUNRISE approach can help ensure compliance with regulatory standards and foster trust in automated vehicles.

UC4.1 focuses on the reverse parking manoeuvre of a truck with a semitrailer. The idea from the application phase is that the sub-UC is a reverse-driving truck with a semitrailer at a logistic hub. Reverse parking manoeuvres for truck-trailer combinations are chalking and have been studied in, e.g., [12] [13] [14] [15] [16] [17]. In the TrustVehicle project, a L3 AD reverse parking truck and trailer combination function was developed [17] targeting the scenario breakdown shown in Figure 34. UC4.1 targets the left branch of this marked in nuances of green colour.

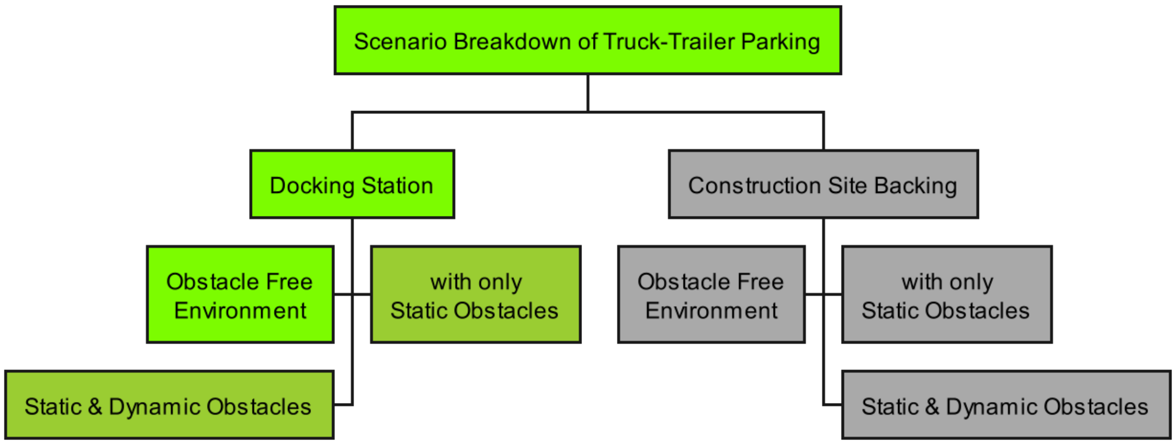

Scenario Breakdown of Truck-Trailer Parking
Docking Station
Construction Site Backing
Obstacle Free Environment
with only Static Obstacles
Static & Dynamic Obstacles
Obstacle Free Environment
with only Static Obstacles
Static & Dynamic Obstacles

Figure 34: Scenario breakdown based on *[17]*. The grey colour is out-of-scope for sub-UC4.1.

The authors of [17] give good motivation for the need for ADAS/AD parking functions summarized in the following bullets:

- Human drivers commonly need several manoeuvres to bring the trailer in the correct position, either to park correctly to the dedicated slot in the docking station or to bring the truck in position on the construction site.
- For both docking station scenarios and construction sites, a main concern is the time spent to position the trailer, but especially for construction sites, is also concerned with surrounding traffic and other road users, such as pedestrians.

3.7.1 Setup

Based on the indicative test scenario described in SUNRISE deliverable D7.1 [1] three preliminary test set-ups have been identified with accompanying Safety Performance Indicators (SPIs). These are shown in Figure 35 and shortly summarized in the following bullets. Note that the set-ups are tentative. It should be seen as indicative and may be changed, e.g., sensor set-up is still to be decided, as well as the layout for the test site.

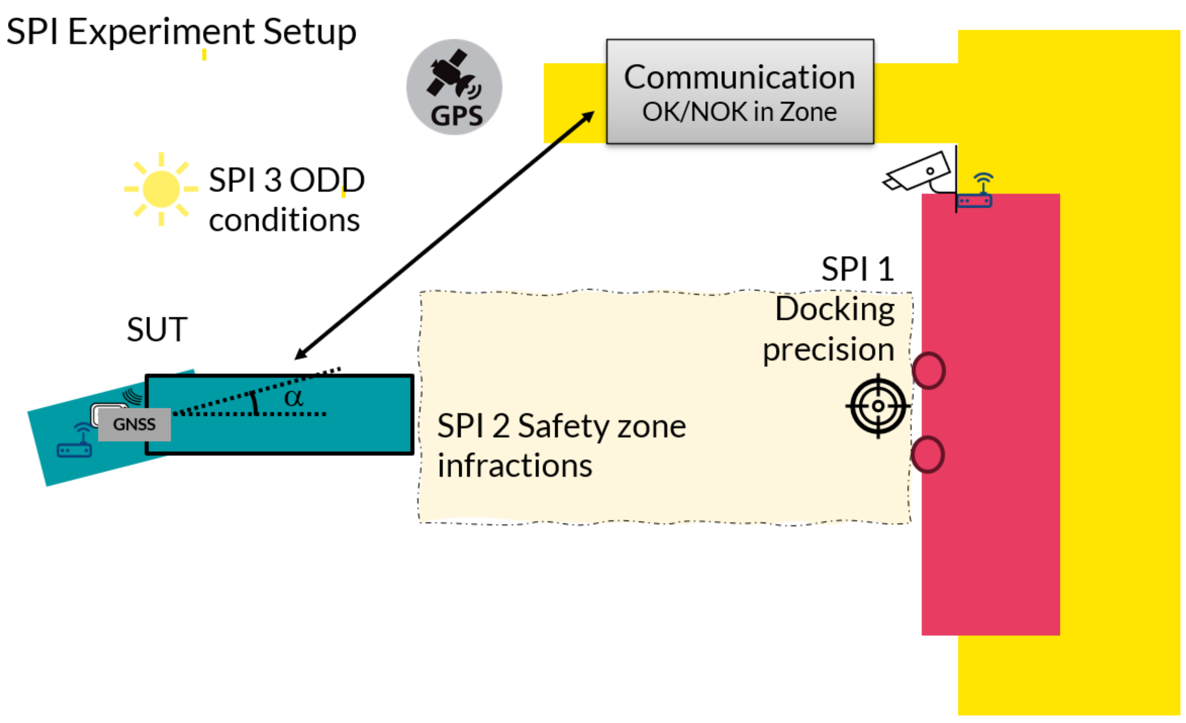


Figure 35: Schematic view of the test set-up connected to the select SPIs.

- ***SPI 1: Docking precision***

  SPI 1 evaluates the docking precision with the semitruck repeatedly starting from the same position. The light condition of the ODD is daylight.

- ***SPI 2: Safety Zone infractions***

  For SPI 2 a safety zone is added in which the truck is expected to move. The starting position is expected to vary, and the test evaluates whether the truck manages to stay inside the safety zone. The light condition of the ODD is considered optimal for the sensors, i.e. no functional insufficiencies.

  The complexity of the test can be increased by extending the ODD to include worse light conditions and evaluating how the semitruck manages to stay in the safety zone while dimming the light.

- ***SPI 3 ODD conditions***

  The intention with SPI 3 is to further evaluate variations in ODD conditions by introducing human beings to the test site and to further evaluate the environmental conditions.

The backing function for the RISE model truck is implemented using WayWise, an open-source rapid prototyping library for connected, autonomous vehicles developed by RISE. A schematic overview of the parts of the WayWise framework is shown in Figure 36. The adoption of for physical testing with a model truck respectively simulation will be described in respectively subchapter.

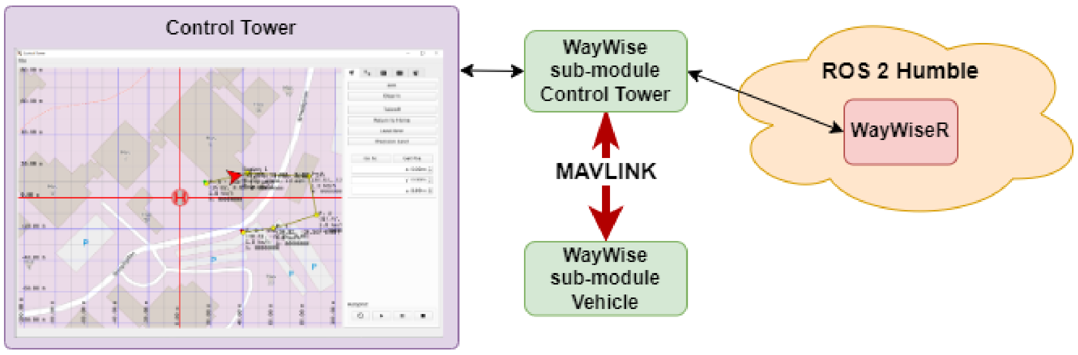


Figure 36: Illustration of the building blocks of the WayWise framework.

There is work related to automated or assistant backing of truck–trailer systems in the literature. Commonly they are for low speed, assuming a simplified linear bicycle model is sufficient, approximating the wheels on an axis to one wheel in the middle and approximating multiple wheel axis to a single axis in the front and back, respectively [18] [19] [20] [21] [22] [23]. Further, no slip is assumed. Commonly they use vehicle position control through feedback of the hitch angle (see Figure 37) based on a linearized system approximation as the target system implemented as P or PI controllers. During the last few years, there has also been an interest in multi-trailer vehicles [24] [25] [26] [27] [28] [29] [16] [30] [31]. Common variants of the pure pursuit algorithm are used [32] for path tracking. For UC4.1, a similar algorithm using the Lyapunov controller [33] [34] is identified to be suitable for the backing function.

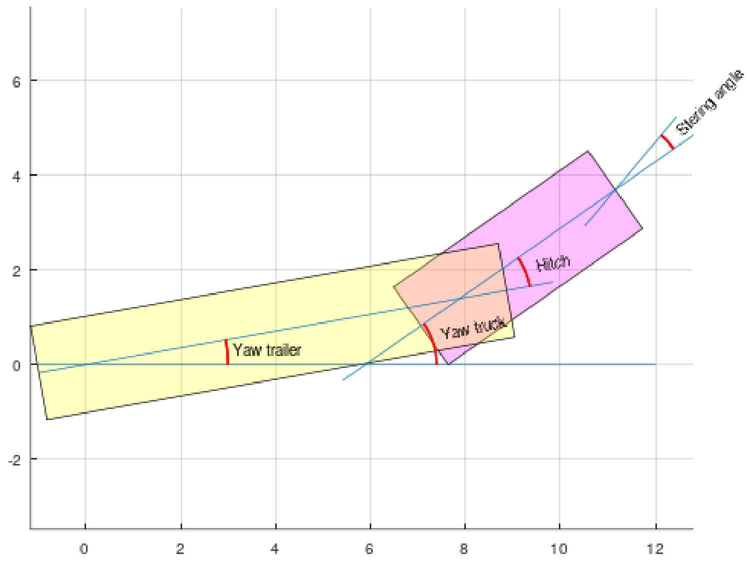


Figure 37: Illustration of a tractor with a connected semitrailer and important angles.

The mathematical model of the backing algorithms together with a simplified kinematic model of the tractor – semitrailer has been implemented using Python and used to evaluate suitable algorithms. Figure 38 shows the calculated trajectories for the back wheel and the trailer in **blue** respectively back wheel of the truck in **yellow** while following the course in **red**. The **green** circle is the calculated turning radius of the trailer, and the **magenta** circle is the look ahead distance. The same colours are valid for the truck and trailer trajectories shown in Figure 39, but now the starting position, including headings, is randomized. In addition, randomized errors are added to the position, yaw and hitch values used by the path-tracking algorithm.

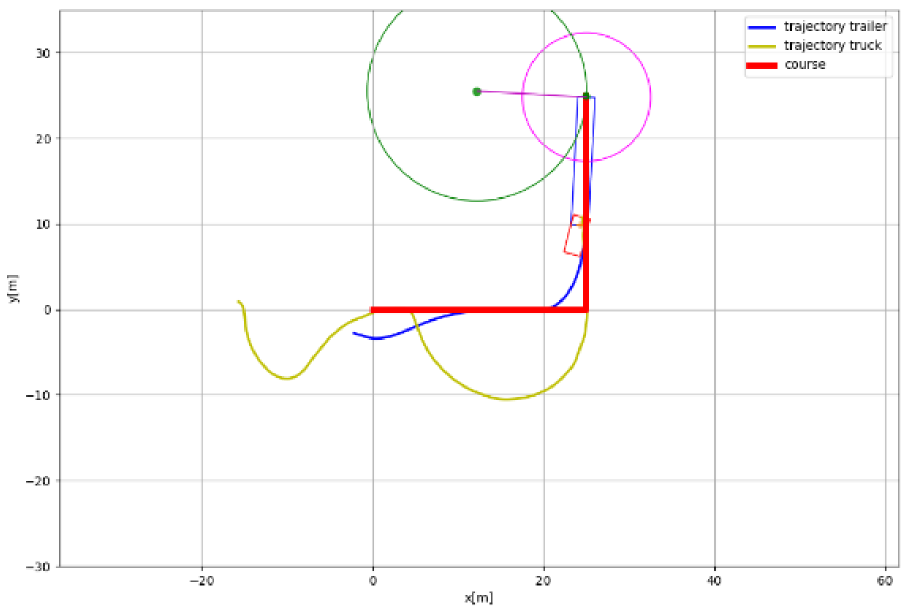


Figure 38: Calculated movements of truck respectively trailer back wheels.

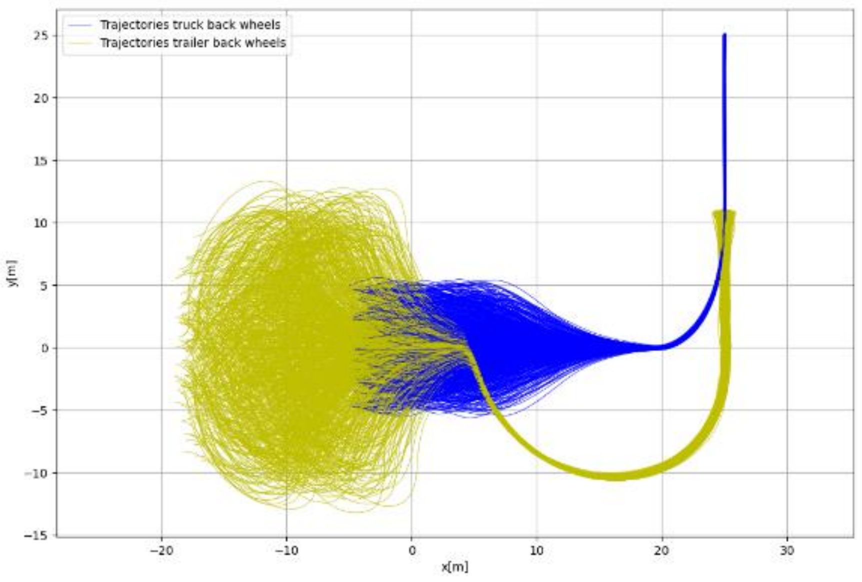


Figure 39: Calculation of trajectories out of randomized start position.

3.7.1.1 Virtual testing

The virtual testing is carried out by RISE using Carla combined with WayWise using the Carla ROS-bridge. A schematic illustration of the simulation set-up to be used is shown in Figure 40.

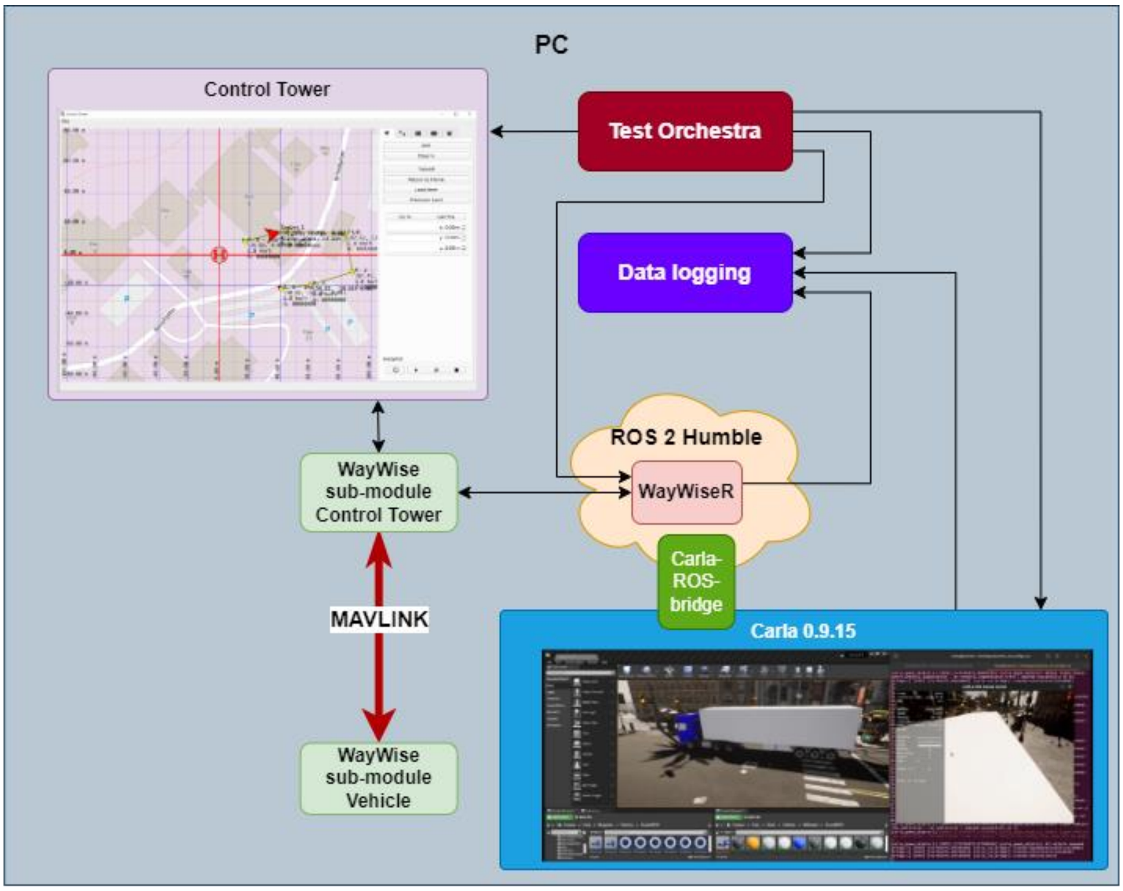


Figure 40: Schematic view of the simulation set-up to be used for sub-UC4.1.

The simulation set-up includes the following parts

- Carla v0.9.15 [35]
- Fork of Carla-ROS-bridge modified to support RISE set-up
- ROS2 Humble [36]
- WayWise/WayWiseR/Control Tower [37]
- These are complemented with software for “Test orchestra” and “Data logging”

3.7.1.2 Proving ground testing

Most of the physical testing is intended to be performed by RISE using a model truck, while a minor part of testing (or demonstration) is done by Chalmers using a full-size truck.

At the time of writing this, a specific logistic hub is not identified, but Chalmers does have contact with partners in other projects that hopefully can allow them to perform tests at an appropriate logistic hub.

Testing using the RISE model truck

The RISE model truck is based on a Tamiya Scania R620 model truck in scale 1:14 combined with a Tamiya semitrailer model in the same scale. The model has been modified to use the WayWise prototyping library to control the truck. A schematic view of the architecture is shown in Figure 41 and the intention is to minimize the differences in the control software between the simulation and the physical tests. Due to limited space and capacity in the onboard computer, parts of the functionality are placed in the Control Tower operated on a PC.

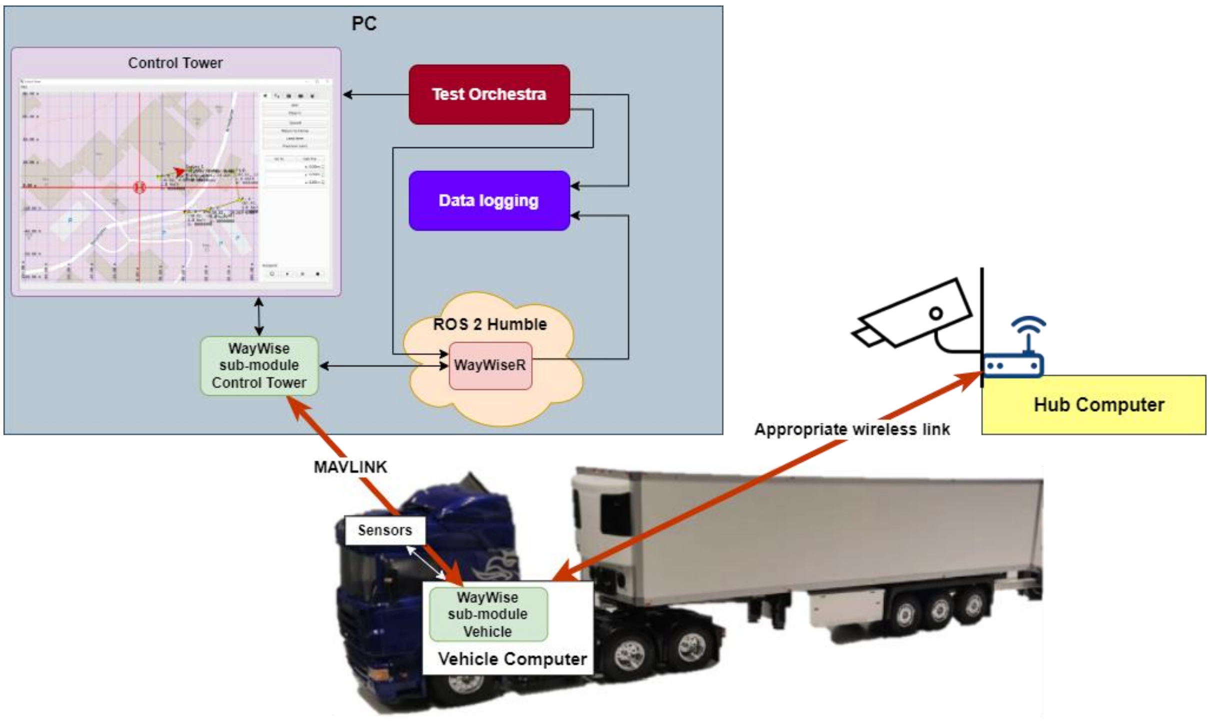


Figure 41: Schematic view of the software set-up with RISE model truck.

Testing using Chalmers full-size truck

Chalmers has access to a Volvo truck and a semitrailer that will be used for SUNRISE. However, it will only be feasible to do a few test runs in the scope of the SUNRISE project. Further, as no production-ready backing function is available in the Chalmers truck, the tests will be performed using a human driver. A large effort is needed to adapt the backing function to the full-size truck, and the economic and safety risks are also considered too high to use the demonstrator backing function developed for the RISE model truck.

3.7.2 Covered aspects of the SAF

The table below shows the contributions to the SAF block validation per partner for UC4.1.

Table 9: Contributions to the SAF validation per Partner in UC4.1

Partner/ Block	CHAL	RISE
SUNRISE DF		

Query & Concretise	x	x
Allocate		x
Execute	x	x
Test Evaluate	x	x
Coverage		
Decide	x	x

The SAF validation process connected to the use case begins with Query & Concretise, where test cases generated are based on full-scale truck data due to the absence of an existing scenario database.

The block distributes test cases to suitable environments, with virtual simulations as the primary focus. Some tests are also allocated to physical testing using a model truck and, in specific cases, a full-size truck. Validate that pre-existing test environment capability can match the test environment requirements.

During the Execution block, virtual and physical tests are carried out, with data collection to ensure the test validity. The Test Evaluate step focuses on confirming the validity of each test and determining its relevance to the safety performance indicators and the use of such in the SAF. Finally, the Decide block investigates the criteria for test completion, determining when sufficient test coverage has been achieved to effectively evaluate the specific safety performance indicators. Each block contributes to systematically validating the SAF.

Query & Concretise

Validate that concretisation provides all the information necessary to allocate tests to an environment. For UC4.1, the involved partners cannot access any existing scenario database. That lack of a source for scenarios is substituted by access to data collection on a full-scale truck.

RISE: A hypothetical query, built on the SUT, ODD, and test objectives, guides the collection of initial scenario data from these experiments. This data then undergoes scenario concretisation to ensure the generated tests are efficient and complete. Validate sufficient coverage of relevant test space and test coverage to be free from critical gaps within the region of interest or across the parameter space. Since an exhaustive search is foreseen to be achieved, the effectiveness of combinatorial testing can be investigated.

Using these concretised data, test cases are generated to evaluate safety performance indicators (SPIs). Each test case specifies a concrete test scenario, including inputs, preconditions, and expected results. **CHAL** will support the validation of the “Query” part in this block by evaluating event and manoeuvre identification approaches as potential instruments to partially compensate for the lack of existing scenario databases.

CHAL: will support the validation of the “Concretisation” part in this block by evaluating the criteria assessing scenario data completeness, data accuracy, and absence of data corruption in combination with test execution (cf. below). The versioning and traceability criterion is planned to be assessed in collaboration with **RISE** during the evaluation of the block “Test Evaluation” for the planned test data collections.

Allocate

RISE: The UC4.1 will use the initial allocation of the test cases to the test instances with the process from D3.3 “Report on the Initial Allocation of Scenarios to Test Instances” [5].

With two or more test instances that can execute comparable tests, evaluating the effectiveness of the test instances allocation is possible.

Execute

RISE: Tests will be performed as virtual tests (D4.4 - Harmonized V&V simulation framework [6]) and physical tests. Efficient orchestration requires test scenarios to be machine-readable and suitable for batch testing. A goal for efficiency is that the recorded data needed to evaluate the validity and results of tests will be collected automatically. Verifying that all results comply with the input required by the “Test Evaluate” and “Coverage” blocks, also confirming that data is free from errors related to test preparation or execution.

CHAL will support the validation of the execution aspect in this block by planning, preparing, and conducting several test data collections with a real-scale truck according to the validation criteria as listed in Section 2.2. In particular focus are concrete aspects such as data presence, data synchronicity, absence of data corruption, and data completeness to meet the expectations for the validation of the block “Test Evaluate”.

Test Evaluate

RISE: UC4.1 aims firstly to evaluate the validity of the test and then the relevance of the result contribution to the safety performance indicators previously described and criteria from the upcoming deliverable D3.5. The UC will validate that it is possible to use distribution-based and rule-based indicators. The results must lend themselves to be returned to the SAF to be aggregated towards the Required test coverage.

CHAL will support the validation for this block by extracting, analysing, and providing meta-information about the executed tests relevant to the validity of the data.

Decide

RISE: Validate the applicability of the guidelines from the upcoming Deliverable D2.3 on the results of the UC.

CHAL will support the validation of the applicability of the guidelines focusing in particular on the evaluation of the several real test scenarios using the real-scale truck to unveil potential improvement aspects with the goal to incorporate the experiences from the tests.

3.7.3 Deviations from D7.1

The deviations from SUNRISE deliverable D7.1 [1] primarily concern the demonstration plans, which have been refined to be more precise and concrete.

3.8 Use case 4.2: Freight vehicle automated parking validation – Truck low-speed connected perception cyber-security

In Use Case 4.2, the SUNRISE SAF is tested for truck low-speed connected perception cyber-testing. The SUT is a connected perception AD subsystem that is compromised by cyber-security threats. The main aim is to combine in a simulation environment several aspects simultaneously (physical environment, perception, V2X connectivity) and study the effects of physical or remotely executed cyber threats on collective environment awareness.

There are two functional scenarios defined from T7.1 for assessing the connected perception cyber-security performance in use case 4.2. The first is the “distorted camera input” scenario, whereas the second one is the “CPM message attacked” scenario.

The use case 4.2 involves the partners ICCS and RISE. ICCS will run simulations in the CARLA environment with a model of the SUT which is an on-board perception receiving a collective perception message from a static infrastructure node equipped with a camera sensor. CPM exchange is simulated via ns-3 simulator environment or emulated spoofed CPMs are created without the use of a network simulator (the focus here is not in the network aspects but in the content of the CP messages). On the other hand, RISE will re-use its controller from UC 4.1 in order to demonstrate the effect of a spoofed CPM to the truck planning algorithm. In addition, RISE will explore their communication-based fault and attack simulation engine (ComFASE) with cybersecurity jamming attacks such as barrage jamming and destructive interference.

Note: The network simulation environment used by ComFASE [38] is OMNET++. However, since this environment is not integrated into CARLA, ICCS will explore the possibility of integrating this attack injection engine in CARLA to be able to perform communication-based attacks in the simulation environment. When it comes to modelling the impact of perception attacks on camera images, ICCS will also explore the possibility of using a CARLA-based fault and simulation engine (CarFASE) [39] developed by RISE to evaluate the test scenarios in the simulation environment. However, since the project focus is not merely on developing new co-simulation environments but contributing to the safety argumentation of complex systems, hence the assumed cybersecurity attacks can be also simulated in a naif way.

3.8.1 Setup

3.8.1.1 Virtual testing

In use case 4.2, virtual testing is carried out by ICCS.

ICCS will use CARLA simulator and CARLA-network co-simulation environment as described in the deliverable D4.3 [9].

The overall ICCS co-simulation framework for the use case 4.2, as specified in SUNRISE deliverable D4.4 [6] (see Figure 42), serves as the foundation for conducting virtual tests, providing a robust and reliable platform for validating the safety of SUT through the SUNRISE SAF.

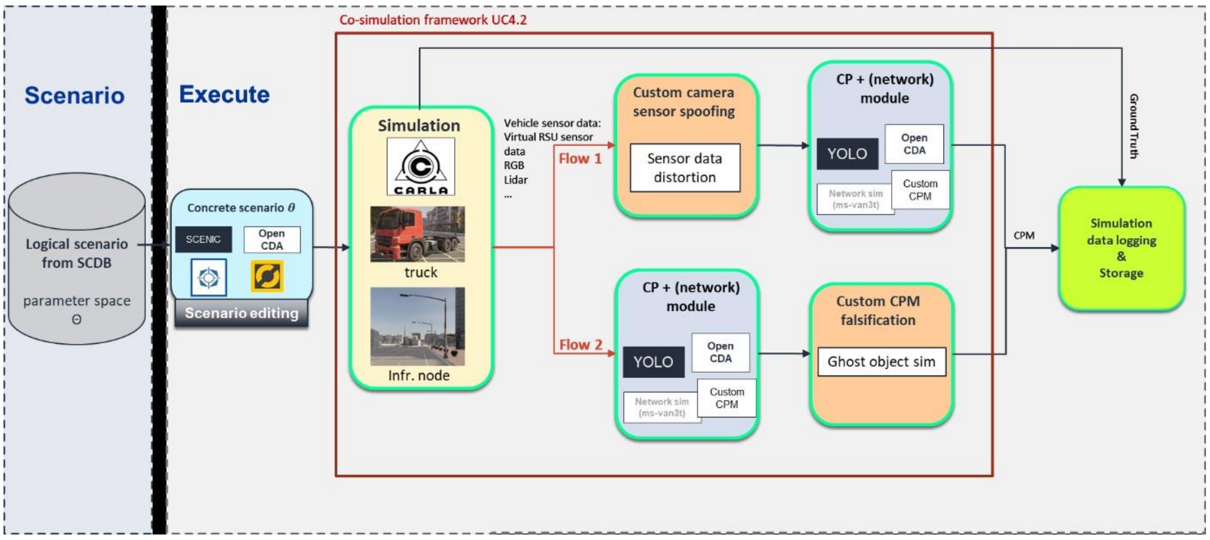


Figure 42: ICCS co-simulation framework for UC 4.2.

The simulation framework carried out by ICCS for use case 4.2 differs from that of use case 1.3 on two fronts, a) it is limited to one agent (truck) and one static infrastructure node and focuses on parking manoeuvre, and b) it includes emulation of two types of cyber-attack on the sensor or V2X message level, to investigate the impact of sensor output or data falsification on the outcome of the manoeuvre.

Following the above, the cybersecurity simulation will be implemented in two ways, first using custom camera sensor spoofing (exploiting light mechanisms inside CARLA simulator) to interfere with the quality of the raw sensor data, and second using CPM falsification by introducing ghost objects inside a CPM generated by the infrastructure node and transmitted to the truck. Both flows are illustrated in Figure 42 and will be investigated either separately or in combination. The produced CPM, raw sensor data and ground truth information will be logged for further processing.

Evaluation of the full AD stack

For evaluating the full AD stack, ICCS will use the controller for the parking manoeuvre implemented by RISE. Otherwise, already existing controllers available in CARLA will be used. The KPIs here are related to safety metrics such as time to collision etc.

3.8.2 Covered aspects of the SAF

The ODD and the two functional scenarios for use case 4.2 are defined in SUNRISE deliverable D7.1 [1]. These are used for generating logical scenarios. The parameters and the ranges are defined internally by WP7. In addition, the KPIs are also defined by WP7.

The use case 4.2 will cover the steps from the definition of logical scenarios until simulation validation. The table below shows the contributions to the SAF block validation per partner for use case 4.2.

Table 10: Contributions to the SAF validation per Partner in UC4.2

Partner/ Block	**ICCS**	**RISE**
SUNRISE DF		
Query & Concretise	x	
Allocate		
Execute	x	x
Test Evaluate	x	x
Coverage		
Decide		

Query & Concretise

Logical scenarios were hand-written as instantiations of the functional scenarios defined in the SUNRISE deliverable D7.1 [1]. Parameterization of these scenarios will be applied and a few concrete scenarios will be created to cover a small subset of the scenario space without attempting to fulfil coverage requirements. This UC will not focus on coverage aspects as this is already covered by the work in UC1.3 (Collective Perception testing)

Allocate

Not supported by the pipeline. All tests are pre-allocated to a virtual environment, i.e. they will be performed using simulation.

Execute

Tests will be performed as virtual tests using a simulation toolchain developed around CARLA. All data that is needed to evaluate the validity and results of tests will be collected automatically. Additionally, all the tests can be run, obtaining the corresponding metrics related to the test objectives.

ICCS: The simulation framework presented in Figure 42 is used to execute the scenarios and the validation criteria from sec. 2.2 will be applied to verify the quality of the simulation execution.

Optional work item (ICCS-RISE synergy): ICCS will integrate a communication network simulator into Carla to be able to then run wireless communication-based cybersecurity tests on the truck backing function. RISE will then support ICCS in the identification and definition of relevant communication-based attacks such as jamming attacks for cybersecurity testing of the communication between the truck and the camera mounted in the logistic hub.

RISE: A key part of this use case is the implementation of a backing function in the simulation environment. RISE will contribute to this use case by developing of a backing function in Carla for the truck (connected to a trailer). This also means that the RISE will contribute to the validation of the execute block by assuring that the backing function has been prepared and executed. Tests will be performed as detailed in D4.4 (Harmonized V&V simulation framework [6]). As part of this, we develop an orchestrator and set points to be used by the truck to perform the backing manoeuvre. The result of the test run, with respect to the backing function, will be logged so that the correctness of the backing function is evaluated in the Test Evaluate block. RISE also contributes to the validation of this block by providing a ground truth execution and logs to be used to check whether log data reveal any potential errors that occurred during test preparation or execution.

Note that as RISE contributes to both use cases 4.1 and 4.2, we plan on exploiting the synergies between these two use cases and support ICCS with validation activities that are circled around the backing function implemented in the Carla environment.

Test Evaluate

ICCS & RISE: Relevant pass/fail criteria will be defined for the truck parking system by RISE and ICCS following guidelines from WP2.Those KPIs are automatically logged in each run along with some metadata which will allow to verify the results' validity through KPIs analysis and visual inspection of the scenario in simulation.

3.8.3 Deviations from D7.1

The deviations from SUNRISE deliverable D7.1 [1] concern the reduction in the size of the demonstration plan of use case 4.2, which is now focusing on the blocks 'Execute' and 'Test Evaluate' of the SAF.

4 CONCLUSIONS

This deliverable builds on the use cases presented and described in SUNRISE deliverable D7.1 [1] and on the current status of the SUNRISE SAF, which is being developed in parallel. The main goal of the SUNRISE Use Cases is the validation and demonstration of the SAF and its blocks from the user perspective, and addressing possible errors, gaps and improvements to the underlying methods, tools and data. The use cases cover a wide range of application areas and involve a variety of different CCAM technologies in order to cover as wide a spectrum as possible. The aim of WP7 is not to validate the CCAM systems themselves.

This document focuses on two main things. The **first part** (chapter 2) briefly **introduces the SAF and its components**, as deliverable D2.3 'Final SUNRISE safety assurance framework' with a detailed description of the framework, will only be published at the end of the project. It also **defines the criteria according to which the validation of the individual blocks of the SAF will be carried out**. This is an important part of the overall validation of the SUNRISE SAF. However, it will not be possible to fully validate the SAF within SUNRISE due to the limited number of use cases. WP7 provides a starting point for full validation of the SAF, but this full validation is outside the scope of the SUNRISE project. The SAF blocks that will be validated are:

SUNRISE DF

The SUNRISE Data Framework (DF) serves as an interface between the "Store" and the "Query & Concretise" components of the SUNRISE SAF. Validation requirements include:

- **User authorisation** for access.
- The DF provides the expected logical scenarios with parameters and ranges or distributions.
- Consistent **query alignment** across formats.

Query & Concretise

This block must provide the complete information for executing tests and collecting data for test evaluation. This is a big block with a number of validation requirements including:

- **Queries** retrieve relevant scenario data, adhering to D5.2 and ensuring consistency, accuracy, reproducibility, and system resilience.
- **Concrete Scenarios** entail accuracy, parameter distribution adherence, sensitivity analysis, parameter dependency checks, and version control.
- Check that all specified **behaviours** conform to the ontology structure defined in D5.2.
- Define a **minimum rate of change** in evaluation metrics between sampling points and validate that this rate of change is not violated.

- Establish a **maximum rate of change** in evaluation metrics between sampling points to avoid under-representation and validate that this rate of change is not violated.

Allocate

The allocation should be done by using the process and metrics of D3.3 "Report on the Initial Allocation of Scenarios to Test Instances" [5]. Validation requirements include:

- Checking the process of D3.3 [5] by evaluating if the **correct test instances and fidelities** have been chosen.
- Checking the process of the upcoming D3.5 by evaluating if the **re-allocation** has been done correctly.

Execute

The Execute block conducts tests in virtual, hybrid, or real-world settings. Validation requirements include:

- check test preparation, execution, and data completeness.
- Error detection during execution relies on log analysis, covering issues like simulation bugs or hardware errors.

Coverage

The coverage must provide information on the coverage of the sample points and regions compared to the whole parameter space of the logical scenarios from SUNRISE DF (see D5.3). Validation requirements include:

- **Compare the coverage** of known parameter spaces to the result of the Coverage block.
- Define region(s) of interest with lower density compared to other regions and **check whether the right feedback was provided**.

Test Evaluate

This block provides the evaluation of the safety performance of the SUT for a single test case and the information on the validity of the test run.

This can be done by **applying safety-related metrics** to conclude the safety performance of the SUT for the single test case.

Decide

This block must provide the safety argumentation of the SUT based on the provided coverage information and test evaluation results. The guidelines are not available at this moment but will be part of D2.3.

Chapter 2 also indicates which use cases will cover which SAF blocks (see Table 1).

The **second part** of this deliverable (chapter 3) **refers to the 8 use cases and describes their design and current development status**. It looks at the current setup of the use cases and the plans for virtual testing, XiL testing and proving ground testing. A lot of this work is the result of WP4 and its latest deliverables D4.3 (“Report on CCAM simulation tool landscape”) [9] and D4.4 (“Report on the Harmonised V&V Simulation Framework”) [6].

The use cases also go into more detail about which validation activities are planned in the individual SAF blocks. Each use case has a section called “**Covered aspects of the SAF**” in which all partners describe their plans to validate the individual SAF blocks. The actual validation of the SAF will be the content of the next deliverable D7.3.

The partners of T7.3 and the people working on the use cases will be the main target of this deliverable since it gives them the ability to carry out the demonstration and validation of the individual blocks of the SUNRISE SAF.

5 REFERENCES

[1] SUNRISE, “D7.1 CCAM Use cases validation requirements,” https://ccam-sunrise-project.eu/deliverable/d7-1-ccam-use-cases-validation-requirements/, 2023.

[2] SUNRISE, “D5.1 Requirement for CCAM safety assessment data framework content,” https://ccam-sunrise-project.eu/deliverable/d5-1-requirements-for-ccam-safety-assessment-data-framework-content/, 2024.

[3] SUNRISE, “D3.2 Report on Requirements on Scenario Concepts Parameters and Attributes,” https://ccam-sunrise-project.eu/deliverable/d3-2-report-on-requirements-on-scenario-concepts-parameters-and-attributes/, 2024.

[4] SUNRISE, “D6.1 Methodology for SCBD application for generic use cases,” 2024.

[5] SUNRISE, “D3.3 Report on the Initial Allocation of Scenarios to Test Instances,” 2024.

[6] SUNRISE, “D4.4 Report on the Harmonised V&V Simulation Framework,” 2024.

[7] “https://www.euroncap.com/media/80156/euro-ncap-aeb-lss-vru-test-protocol-v451.pdf,” [Online].

[8] “https://www.asam.net/standards/detail/osi/,” [Online].

[9] SUNRISE, “D4.3 Report on CCAM simulation tool landscape,” 2024.

[10] “https://5gcarmen.eu,” [Online].

[11] E. W. G. “. a. A. Driving”, “Connected, Cooperative and Automated Mobility Roadmap,” ERTRAC, 2024.

[12] G. Kıvançlı, *Auto-Trailer Parking Project & HIL Studies,* 2020.

[13] Y. Hamaguchi and P. Raksincharoensak, “Automated steering control system for reverse parking maneuver of semi-trailer vehicles considering motion planning by simulation of feedback control system,” *Journal of Robotics and Mechatronics,* vol. 32, no. 3, 2020.

[14] TrustVehicle, “Improved trustworthiness and weather-independence of conditional automated vehicles in mixed traffic scenarios,” Fact Sheet H2020.

[15] “TrustVehicle,” [Online]. Available: https://www.trustvehicle.eu/. [Accessed 05 September 2024].

[16] O. Ljungqvist, “Motion Planning and Stabilization for a Reversing Truck and Trailer System,” 2015.

[17] A. C. Manav, I. Lazoglu and E. Aydemir, “Adaptive Path-Following Control for Autonomous Semi-Trailer Docking,” *IEEE Transactions on Vehicular Technology,* vol. 71, no. 1, 2022.

[18] M. Sampei, T. Tamura, T. Itoh and M. Nakamichi, “Path tracking control of trailer-like mobile robot,” in *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, 1991.

[19] C. Pradalier and K. Usher, “A simple and efficient control scheme to reverse a tractor-trailer system on a trajectory,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2007.

[20] C. Pradalier and K. Usher, “Robust trajectory tracking for a reversing tractor trailer,” *Journal of Field Robotics,* 2008.

[21] J. L. Martínez, M. Paz and A. García-Cerezo, “Path tracking for mobile robots with a trailer,” in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 2002.

[22] V. Cviklovic, R. Srnanek, D. Hruby and M. Harnicarova, “The Control Reversing Algorithm for Autonomous Vehicles with PSD-Controlled Trailers,” *Acta Technologica Agriculturae,* vol. 24, no. 4, 2021.

[23] H. Shin, M. J. Kim, S. Baek, C. D. Crane and J. Kim, "Perpendicular Parking Path Generation and Optimal Path Tracking Algorithm for Auto-parking of Trailers," *International Journal of Control, Automation and Systems,* vol. 20, no. 9, 2022.

[24] C. Altafini, A. Speranzon and B. Wahlberg, "A feedback control scheme for reversing a truck and trailer vehicle," *IEEE Transactions on Robotics and Automation,* vol. 17, no. 6, 2001.

[25] C. Altafini, A. Speranzon and K. H. Johansson, "Hybrid control of a truck and trailer vehicle," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2002.

[26] E. P. Biever, "A nonlinear path tracking approach for reversing an A-double truck," Technische Universiteit Eindhoven, 2023.

[27] N. Evestedt, O. Ljungqvist and D. Axehill, "Path tracking and stabilization for a reversing general 2-trailer configuration using a cascaded control approach," in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2016.

[28] T. Ghandriz, B. Jacobson, P. Nilsson, L. Laine and N. Fröjd, "Computationally efficient nonlinear one- A nd two-track models for multitrailer road vehicles," *IEEE Access,* vol. 8, 2020.

[29] O. Holmer, "Motion Planning for a Reversing Full-Scale Truck and Trailer System," Linköping University, Automatic Control, 2016.

[30] O. Ljungqvist, "On motion planning and control for truck and trailer systems," Linköping University, Department of Electrical Engineering, Automatic Control., Linköping, 2019.

[31] P. Nilsson and K. Tagesson, "Single-track models of an A-double heavy vehicle combination," Chlamers University of Technology, Göteborg, 2014.

[32] R. C. Coulter, "Implementation of the Pure Pursuit Path Tracking Algorithm," Carnegie Mellon University, 1992.

[33] A. Elhassan, "Autonomous driving system for reversing an articulated vehicle," The Royal Institute of Technology, Department of Automatic Control, Stockholm, 2015.

[34] K. Kvarnfors, "Motion Planning for Parking a Truck and Trailer System," KTH, School of Electrical Engineering and Computer Science (EECS), Stockholm, 2019.

[35] A. Dosovitskiy, G. Ros, F. Codevilla, A. M. López and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *Conference on Robot Learning*, 2017.

[36] S. Macenski, T. Foote, B. Gerkey, C. Lalancette and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics,* vol. 7, no. 66, 2022.

[37] M. Damschen, R. Häll and A. Mirzai, "WayWise: A rapid prototyping library for connected, autonomous vehicles," *Software Impacts,* vol. 21, p. 100682, 1 9 2024.

[38] "https://github.com/RISE-Dependable-Transport-Systems/ComFASE," [Online].

[39] "https://github.com/RISE-Dependable-Transport-Systems/CarFASE," [Online].