SUNRISE

SAFETY ASSURANCE FRAMEWORK FOR CONNECTED, AUTOMATED MOBILITY SYSTEMS

D3.5
Report on the Dynamic Allocation and Validation of Test Runs

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	WP3: CCAM V&V Methodology for Safety Assurance
Deliverable number	D3.5: Report on the Dynamic Allocation and Validation of Test Runs
Deliverable responsible	Jason Xizhe Zhang, University of Warwick
Status - Version	Final – V1.0
Submission date	27/08/2025
Keywords	Scenario, Safety Assurance, Requirements, Interfaces, CCAM system, Test run validation, Dynamic allocation, Test instances, Test environments, Test Run Validation Metrics, Test Evaluate Metrics, Fidelity

Authors

Role	Name
Main author:	Jason Xizhe Zhang (UoW)
Contributing authors:	Thaddaeus Menzel (IDI), Anders Thorsén (RISE), Bernhard Hillbrand (VIF), Mohammed Shabbir Ali (VEDECOM), Jobst Beckmann (ika)

Quality Control

	Name	Organisation	Date
Peer review 1	Marcos Nieto Doncel	Vicomtech	01/07/2025
Peer review 2	Stefan de Vries	IDIADA	18/07/2025

Version history

Version	Date	Author	Summary of changes
0.1	10/01/2025	Jason Zhang	Started initial draft
0.9	01/08/2025	All authors	First complete draft available and reviewed
1.0	27/08/2025	Jason Zhang	Final version

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS AND ACRONYMS

Abbreviation	Meaning
AD	Automated Driving
ADS	Automated Driving System
AEB	Autonomous Emergency Braking
ASAM	Association for Standardization of Automation and Measuring Systems
CCAM	Connected, Cooperative, and Automated Mobility
COTSATO	COncretizing Test Scenarios and Associating Test Objectives
DoE	Design of Experiments
ECU	Electronic Control Unit
FMI	Functional Mock-up Interface
GPU	Graphics Processing Unit
HiL	Hardware-in-the-Loop
ISMR	In-Service Monitoring and Reporting
KPI	Key Performance Indicator
LHS	Latin Hypercube Sampling
NATM	New Assessment/Test Method for Automated Driving
ODD	Operational Design Domain
OEM	Original Equipment Manufacturer
OSI	Open Simulation Interface
PDF	Probability Density Function
SAF	Safety Assurance Framework
SCDB	SCenario DataBase
SUNRISE	Safety assUraNce fRamework for connected, automated mobIlity SystEms
SUT	System Under Test
TTC	Time To Collision

UC	Use Case
V&V	Verification and Validation
ViL	Vehicle-in-the-Loop
WP	Work Package

EXECUTIVE SUMMARY

Safety assurance of Cooperative, Connected, and Automated Mobility (CCAM) systems is a crucial factor for their successful adoption in society, yet it remains a significant challenge. It is generally acknowledged that for higher levels of automation, the validation of these systems by conventional test methods would be infeasible. Furthermore, certification initiatives worldwide struggle to define a harmonized safety assurance approach enabling massive deployment of CCAM systems.

The **SUNRISE** project develops and demonstrates a **CCAM Safety Assurance Framework** (SAF). The overall objective of the SUNRISE project is to accelerate the large-scale and safe deployment of CCAM systems. In alignment with international twin projects and initiatives, the project aims to achieve this objective by providing a SAF consisting of three main components: a Method, a Toolchain and a Data Framework. The **Method** is established to support the SAF safety argumentation, and includes procedures for scenario selection, sub-space creation, dynamic allocation to test instances and a variety of metrics and rating procedures. The **Toolchain** contains a set of tools for safety assessment of CCAM systems, including approaches for virtual, hybrid and physical testing. The **Data Framework** provides online access, connection and harmonization of external Scenario Databases (SCDBs), allowing its users to perform query-based extraction of safety relevant scenarios, allocation of selected scenarios to a variety of test environments, and reception of the test results.

This deliverable presents a **method** (hereafter referred to as the "method") **for validating test runs and dynamically allocating test cases** within the SUNRISE SAF. It addresses the challenge of validating individual test runs, and if necessary, to conclude such test runs by dynamically re-matching test scenarios to suitable test instances, considering both scenario requirements and the capabilities of available test environments (as explained in deliverable D3.3 [1]). The objective is to ensure that each test run meaningfully realises the scenario it is supposed to represent and that the evidence produced is trustworthy, relevant and valid for safety assessment purposes. Without the validation of test run, the outcome could still meet safety goals, however such outcome might not be trustworthy due to invalidations at the test run execution level. It is therefore a **key objective** of this deliverable to propose considerations for carrying out validation process of test runs.

At the core of this method is a set of **test run validation metrics**, and a mechanism that links test requirements (often derived from logical scenarios or safety goals and test objectives) to concrete combinations of testing tools, configurations, and system capabilities. This allocation process considers multiple constraints, including test system limitations, scenario triggering conditions, and coverage goals. Importantly, it is designed to be iterative and adaptive, refining allocations based on previous test outcomes and the degree to which scenarios were successfully tested.

To support this, the method includes **test run validation procedures** to assess whether a test instance is capable of fulfilling the intended scenario, and whether it did so during the execution. This includes checks at both the configuration level and the behavioural level,

allowing identification of mismatches, partial realisations, or under-tested conditions. Where gaps or issues are found, feedback is incorporated into a refinement loop that updates capability descriptions, scenario logic, or allocation criteria. By embedding this validation and dynamic allocation process into the SAF, the method helps to ensure that testing is trustworthy, effective, efficient and meaningful. It avoids unmeaningful test outcomes due to impacting factors associated with the test instances, rather than the decisions of the CCAM system.

1 INTRODUCTION

1.1 Project introduction

CCAM systems need to demonstrate reliability in all driving scenarios, requiring robust safety argumentation. It is 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 for cost efficiency reasons.

Worldwide, several initiatives have started to develop test and assessment methods for Automated Driving (AD) functions. These initiatives already transitioned from conventional validation to a scenario-based approach and combine different test instances (physical and virtual testing) to avoid the million-mile issue.

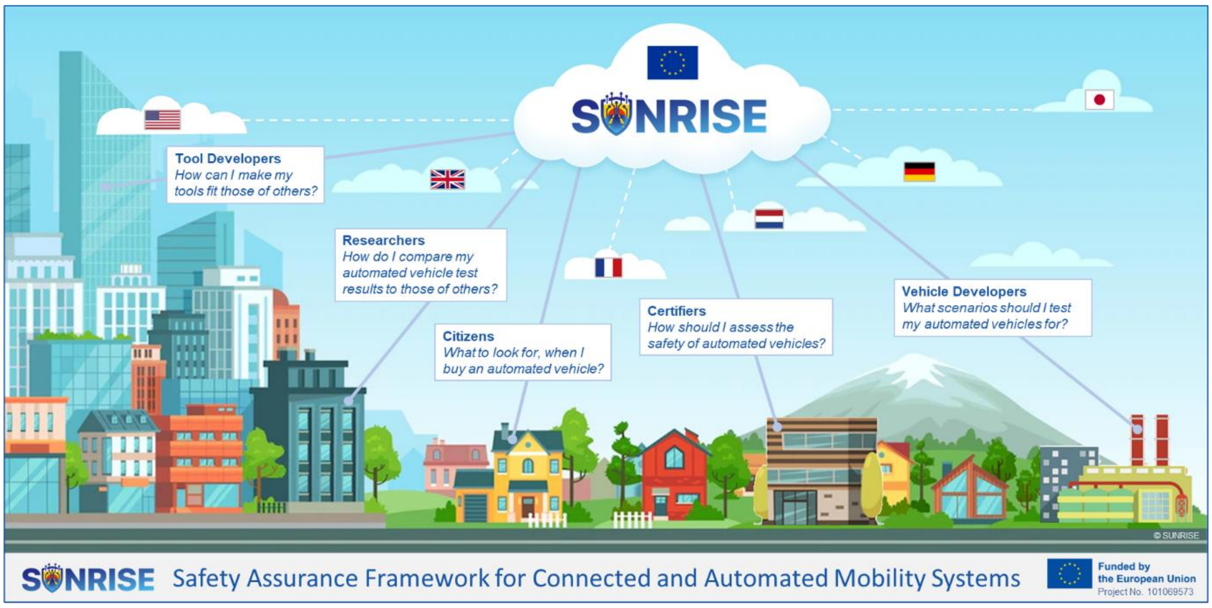
The initiatives mentioned above, provide new approaches to CCAM validation, and many expert groups formed by different stakeholders, are already working on CCAM systems' testing and quality assurance. Nevertheless, the lack of a common European validation framework and homogeneity regarding validation 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 entire 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 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 project initiatives as a baseline, it becomes necessary to move to the next level in the development 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 product of the SUNRISE project. As the following figure indicates, it takes a central role, fulfilling the needs of different automotive stakeholders that all have their own interests in using it.


SUNRISE
Tool Developers
How can I make my tools fit those of others?
Researchers
How do I compare my automated vehicle test results to those of others?
Citizens
What to look for, when I buy an automated vehicle?
Certifiers
How should I assess the safety of automated vehicles?
Vehicle Developers
What scenarios should I test my automated vehicles for?
© SUNRISE
SUNRISE Safety Assurance Framework for Connected and Automated Mobility Systems
Funded by the European Union
Project No. 101069573
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 objective by providing a SAF consisting of three main components: a Method, a Toolchain and a Data Framework. The **Method** is established to support the SAF safety argumentation, and includes procedures for scenario selection, sub-space creation, dynamic allocation to test instances and a variety of metrics and rating procedures. The **Toolchain** contains a set of tools for safety assessment of CCAM systems, including approaches for virtual, hybrid and physical testing. The **Data Framework** provides online access, connection and harmonization of external Scenario Databases (SCDBs), allowing its users to perform query-based extraction of safety relevant scenarios, allocation of selected scenarios to a variety of test environments, and generation of the test results. The SAF will be put to the test by a series of **Use Cases demonstrations**, designed to identify and solve possible errors, gaps and improvements to the underlying methods, tools and data.

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.

The following figure shows the general workplan of the SUNRISE project.

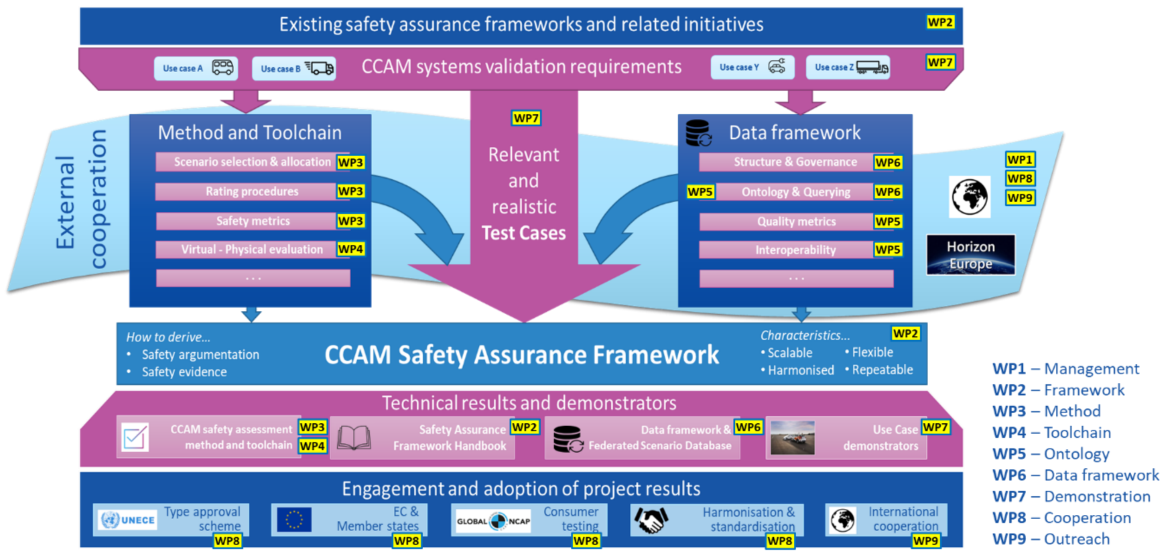

Existing safety assurance frameworks and related initiatives
WP2
Use case A
Use case B
CCAM systems validation requirements
Use case Y
Use case Z
WP7
External cooperation
Method and Toolchain
Scenario selection & allocation WP3
Rating procedures WP3
Safety metrics WP3
Virtual - Physical evaluation WP4
...
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
Data framework & Federated Scenario Database 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: Workplan of the SUNRISE Project

1.2 Purpose of the deliverable

Work Package 3's part in SUNRISE is to define and condense an overall CCAM Verification and Validation (V&V) methodology to support the safety argumentation based on data- and knowledge-driven, scenario-based testing.

The purpose of this deliverable is to **assess the initial test instance allocation** established in deliverable **D3.3** [1] via a **test run validation process**, and to propose **refinements** that improve the accuracy and realisability of the test instance allocation, and consequently to ensure trustworthiness of the overall safety assurance outcome. The dynamic allocation method proposed in this deliverable, builds upon the initial allocation method proposed in deliverable D3.3 [1], by incorporating insights gained through initial stage test planning, toolchain deployment, and the initial execution of selected test runs across simulation and physical environments.

The initial allocation in deliverable D3.3 [1] served as a structured and methodologically grounded mapping between abstract or logical scenarios and suitable test instances. It provided a baseline strategy for distributing scenarios across test instances, such as simulation, proving ground, and on-road, based on their characteristics and the intended validation objectives. As outlined in the scope of WP3, this was an essential step in operationalising the Safety Assurance Framework (SAF), more specifically the **'Allocate'** block.

This deliverable addresses the next step in that process. Building on the foundation of deliverable D3.3 [1] on initial test instance allocation, and based on result from the initial execution, deliverable D3.5 **evaluates** how well the **initial allocation decisions** perform under different model and test instance fidelities. It **spots** any critical discorrelations across lower and higher fidelity models that could invalidate the test outcome produced in the lower fidelity test instance. It **identifies** cases where initial assumptions may require **revision**, where the maturity of specific tools or environments affects the **trustworthiness** of the test outcome, or where additional clarity on performance requirements suggests a different allocation might better serve the safety argument.

By introducing targeted re-allocation suggestions and rationales, this deliverable **enhances** the trustworthiness of test results, and **refines** the knowledge of the scenario requirements and test instance capabilities, thereby **improving** the future test instance allocation process. It helps ensure that test allocation decisions are **trustworthy, valid, technically justified**, and aligned with both system-level verification goals and real-world execution capabilities. Ultimately, D3.5 supports more confident progression toward test execution, by refining the scenario allocation to maximise its practical value and assurance strength.

1.3 Intended audience

This deliverable serves multiple stakeholders. It is intended for a range of stakeholders involved in the planning, execution, and oversight of scenario-based safety assurance activities within the SUNRISE project and beyond.

First and foremost, it is targeted at CCAM safety **test engineers** responsible for the implementation of scenario-based validation activities across simulation, proving ground, and real-world testing environments. The re-allocation mechanism provided in this document are expected to inform their decisions when selecting and configuring test instances in line with evolving system requirements, toolchain fidelity, and test instances capabilities. Secondly, the report is relevant to vehicle safety bodies (such as certifiers, type approval authorities and regulators), who may use it to understand how scenario-to-test allocation decisions are justified, revisited, and adapted as part of a structured and transparent assurance workflow. By documenting the rationale behind re-allocation choices and the traceability to safety performance objectives, this deliverable improves the trustworthiness and relevance of test results of the SUNRISE SAF.

In addition, the contents of this deliverable may be of interest to other parties involved in the SAF workflow, including **scenario designer**, **scenario database maintainers**, and **safety analysis engineers**, who can reference the methodological approach to scenario allocation as a foundation for structured test planning, cross-tool compatibility, and safety assurance traceability. Finally, **project partners involved in other WPs** may find this deliverable useful for aligning test execution activities with the refined allocation strategy presented here.

1.4 Deliverable structure and relation to other parts of project

This deliverable is structured to provide a clear and traceable view of the rationale, method, and results of the test run validation and re-allocation process for scenario-based test within the SUNRISE SAF. More specifically, it interacts heavily with the SAF blocks Allocate, Execute and Test Evaluate. It builds directly on the initial allocation reported in D3.3 [1] and refines that baseline by incorporating practical insights gathered from early execution and planning activities.

The deliverable begins with a contextual overview of the scenario allocation task and a restatement of key SAF elements relevant to re-allocation (Section 2), followed by a high level workflow (Section 3) and a summary of the initial allocation mechanism and how the re-allocation interacts with it (Section 4). The main body of the report (Sections 5) presents the methods for test run validation and dynamic allocation (= re-allocation) of the test cases to test instances, including the feedback loop of possible refinements in the test environment. Section 6 concludes the deliverable, highlighting the key findings and indicating potential future work.

This deliverable is closely related to the following work packages and deliverables:

- **WP3 / D3.3 – Initial Allocation of Scenarios to Test Instances** [1]: D3.3 established the initial mapping between scenarios and test environments using early inputs from the SAF and CCAM system definition. D3.5 does not replace D3.3, but builds upon it by assessing the initial allocation in light of the obtained test results. The scope covered in this deliverable (D3.5) takes place *after* the initial allocation and test run. It firstly validates the test run. Based on the outcome of this validation, it then performs the necessary dynamic allocation of the test case to a different fidelity test instance, and refines the knowledge of the capabilities associated with each of the available test instances.

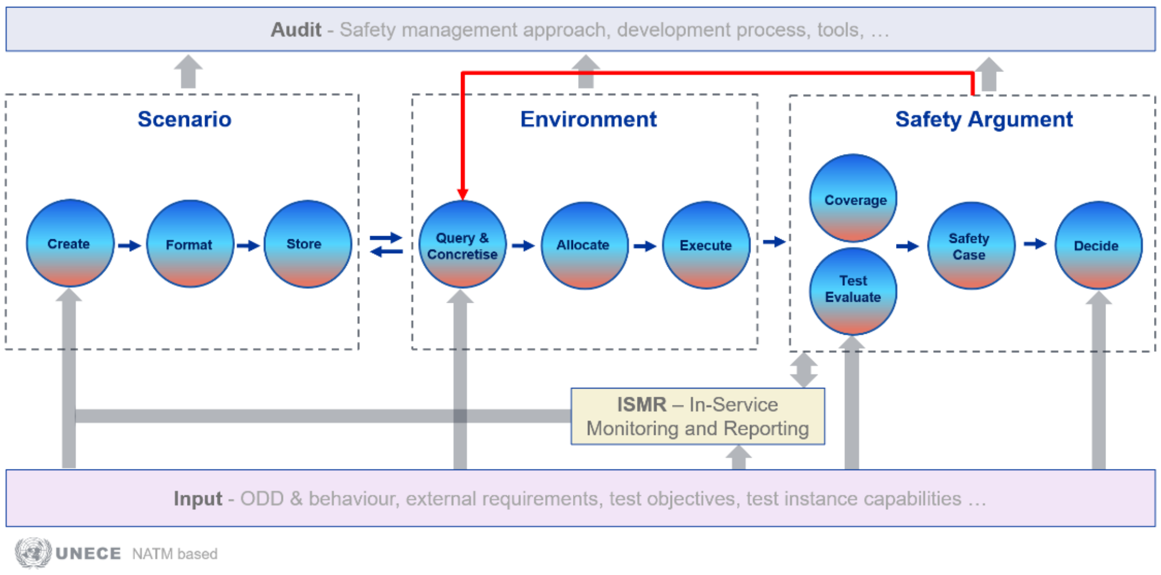
- **WP2 – CCAM Safety Assurance Framework**: The re-allocation work is underpinned by principles and blocks defined within the SAF, particularly those concerning test case allocation (Allocate block), test execution (Execute block), and test evaluation (Test Evaluate block). Based on the initial allocation and test execution, the test outcome becomes accessible to the re-allocation method (which sits within the Allocate block) via the feedback arrow from Safety Argument to Environment.

- **WP4 – CCAM V&V Framework**: Feedback from WP4 activities, such as virtual test framework, integration at the system and sub-system level of the SUT, characteristics of virtual test environment at different fidelity levels, directly influenced the re-evaluation of allocation feasibility. Deliverable D3.5 provides refined means to assess test instance capabilities for more precise test planning.

- **WP8 – Standards and Regulatory Alignment**: The processes and considerations documented for both the test run validation and dynamic allocation, could contribute to standardisation activities. They are relevant from both the developer's perspective (i.e., ensure trustworthy test outcome), and the regulatory body's perspective (able to apply test run validation process to assess test outcome).

2 BACKGROUND AND RELATION TO SAF

The SUNRISE Safety Assurance Framework (SAF) provides a harmonised structure to assess whether a Connected, Cooperative, and Automated Mobility (CCAM) system meets predefined safety requirements. The schematics of the SAF workflow is illustrated in the figure below.



Audit - Safety management approach, development process, tools, …
Scenario
Create
Format
Store
Environment
Query & Concretise
Allocate
Execute
Safety Argument
Coverage
Test Evaluate
Safety Case
Decide
ISMR – In-Service Monitoring and Reporting
Input - ODD & behaviour, external requirements, test objectives, test instance capabilities …
UNECE NATM based

Figure 3. Overview of the SUNRISE Safety Assurance Framework.

It builds on the UNECE's NATM multi-pillar model and integrates scenario-based testing, structured safety argumentation, and data-driven and knowledge-based assessment to support system evaluation from development through deployment. The SAF includes four pillars: Audit, Input, In-Service Monitoring and Reporting (ISMR), and Safety Performance Assurance, the main focus of SUNRISE. This Performance Assurance pillar is structured around three subcomponents: Scenario, Environment, and Safety Argument.

The **Scenario component** manages the creation, formatting and storage of test scenarios critical for CCAM safety validation. Scenarios are created using both data-driven methods [2] and knowledge-based techniques [3], and are stored in individual scenario databases (SCDBs). These scenarios are formatted into different abstraction levels, ranging from functional to concrete, using standardised formats such as ASAM OpenSCENARIO [4] and BSI Flex 1889 [5]. The SUNRISE Data Framework connects multiple SCDBs to enable consistent, federated access while respecting individual SCDBs.

The **Environment component** operationalises test scenarios through 3 blocks: querying and concretising, test environment allocation, and execution. Scenarios are retrieved using ODD descriptions, and behavioural requirements. These scenarios are then allocated to appropriate test instances, ranging from virtual simulations to hybrid or physical testbeds, by matching test case requirements against the capabilities of available test instances. **Key features within the Allocate block are the test run validation and dynamic allocation mechanism** presented in this deliverable, which are triggered via a feedback loop from the Test Evaluate block (see red arrow in figure above). If a test run reveals that the intended scenario was not realised or failed to meet test run validation criteria, the re-allocation attempts to execute the test case within a higher fidelity test instance and examine the correlation. It then updates either the test instance capabilities or the test case requirements based on the findings. In those methods, the safety confidence metric is established probabilistically. The correlation between a higher and a lower fidelity test instance also includes the correlation found between virtual tests and physical test. The Execute block then runs the (re-)allocated scenarios in the designated environment, collecting relevant data for analysis. Together, these steps form an adaptive, structured approach for executing scenarios within the SAF.

The **Safety Argument** component ensures robust safety assessment through four stages. Coverage analyses how well scenarios span parameter spaces and ODD conditions. Test Evaluate verifies if safety criteria (e.g., collision avoidance) are met in each execution. Safety Case compiles structured evidence, validating tools and methods against legal and regulatory expectations. Decide integrates all results to deliver a final safety assurance outcome including a binary safety decision, forming the basis for certification or deployment decisions.

3 HIGH LEVEL WORKFLOW

The figure below illustrates the high level workflow (including the feedback loop) of the test instance allocation mechanism, it contains both the initial allocation (described in deliverable D3.3 [1]) and the re-allocation (described in deliverable D3.5). It can be seen that at this level, the building blocks consist of 'Input', 'Initial allocation & execution results', 'Dynamic allocation', 'SUT requirements', 'Input refinement', and 'Dynamic allocation outcome'. Both the outcome from D3.3 [1] initial test instance allocation, and the outcome from D3.5 validation of test run and dynamic allocation, form the complete test instance allocation method of SUNRISE. Within this overall allocation method, the core outcome from D3.5 is highlighted in boxes with green colour.

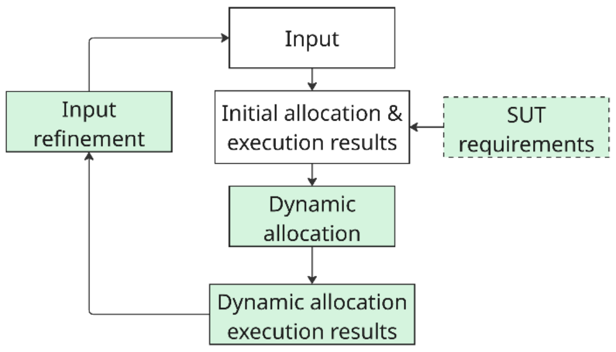

Input
Input refinement
Initial allocation & execution results
SUT requirements
Dynamic allocation
Dynamic allocation execution results

Figure 4. High level workflow of the test instance allocation method

The input and initial allocation boxes are both documented in deliverable D3.3 [1]. The Input layer to the initial allocation contains requirements derived from the test cases, and the known capabilities of the available test instances. The initial allocation contains a workflow that compares the requirements and the capabilities to then allocate the scenario to the most suitable test instance, and consequently execute the scenario within that test instance. The scope of D3.5 then carries on based on the initial execution outcome, passes it through the re-allocation checking workflow to determine whether the test outcome needs double checking, and if it does then the same scenario will be executed within a high fidelity test instance to validate the outcome obtained at the initial test instance allocation and execution. Based on the outcome from this double checking, the known capabilities and test case requirements extraction process might be refined, together with the SUT associated requirements.

4 INTERACTION BETWEEN INITIAL ALLOCATION AND RE-ALLOCATION

The figure below displays the detailed workflow based on the high level workflow illustrated in Figure 4. It is further explained in the next chapter. The additional components to the initial allocation method from deliverable D3.3 [1] include: SUT requirements, Compare SUT requirements with test instance capabilities, Capabilities & requirements refinement. With the dynamic allocation method from this deliverable D3.5, the components include: Scenario realization feasibility, safety confidence evaluation, executing in higher fidelity test instance, correlation between high and low fidelity test instance evaluation, experts reasoning on the correlation.

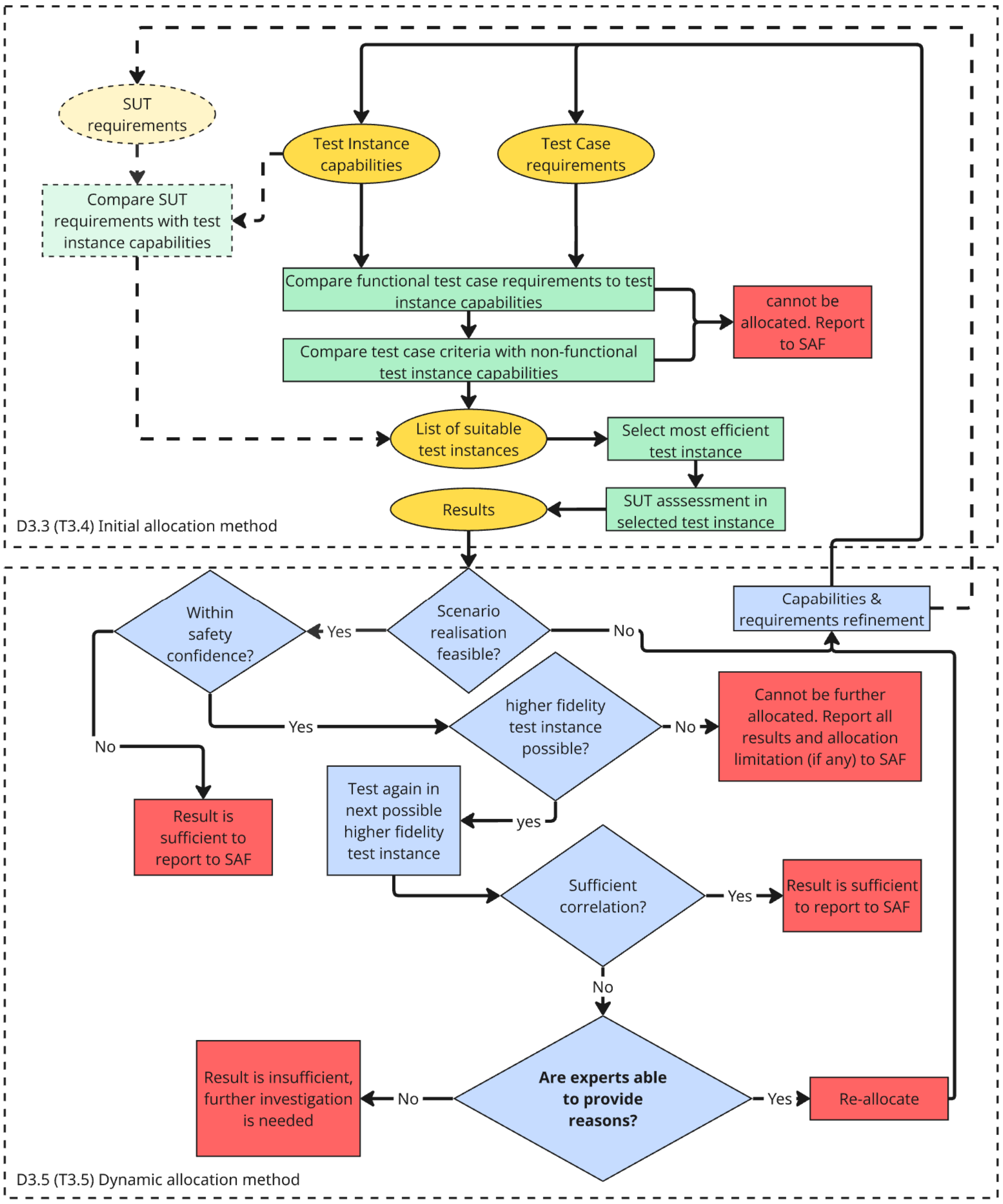

SUT requirements
Test Instance capabilities
Test Case requirements
Compare SUT requirements with test instance capabilities
Compare functional test case requirements to test instance capabilities
cannot be allocated. Report to SAF
Compare test case criteria with non-functional test instance capabilities
List of suitable test instances
Select most efficient test instance
Results
SUT asssessment in selected test instance
D3.3 (T3.4) Initial allocation method
Within safety confidence?
Yes
Scenario realisation feasible?
No
Capabilities & requirements refinement
Yes
higher fidelity test instance possible?
No
Cannot be further allocated. Report all results and allocation limitation (if any) to SAF
No
Result is sufficient to report to SAF
Test again in next possible higher fidelity test instance
yes
Sufficient correlation?
Yes
Result is sufficient to report to SAF
No
Result is insufficient, further investigation is needed
No
Are experts able to provide reasons?
Yes
Re-allocate
D3.5 (T3.5) Dynamic allocation method

Figure 5. Detailed workflow of the test instance allocation method, including both initial allocation and dynamic allocation

Allocation of test cases to test instances is an iterative process involving initial allocation and re-allocation. In this section, the interaction of initial allocation and re-allocation are described.

4.1 Inputs for the initial allocation

The inputs for the initial allocation are the test case requirements and test instance capabilities as described in D3.3 [1] and shown in the image below of the allocation process flowchart. However, some of the SUT requirements important for the test instance allocation process could not be covered by the test case requirements, especially if these requirements are not related to the ODD (more details can be found in section 4.1.3).

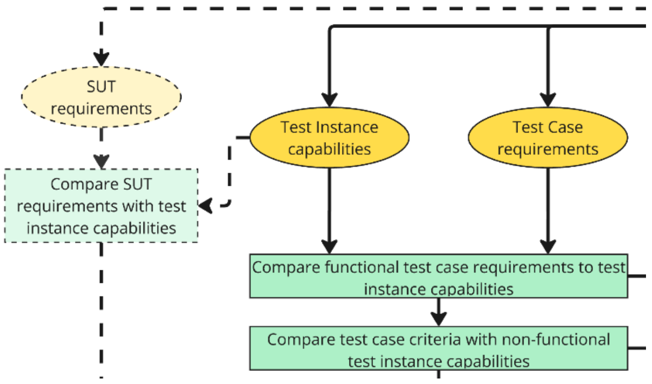

SUT requirements
Test Instance capabilities
Test Case requirements
Compare SUT requirements with test instance capabilities
Compare functional test case requirements to test instance capabilities
Compare test case criteria with non-functional test instance capabilities

Figure 6. input to the initial allocation scheme, plus T3.5 additions to the initial allocation

4.1.1 Test Case Requirements

The test case requirements are defined in the initial allocation process in Deliverable D3.3 [1]. The structure of the test case requirements remains the same for the re-allocation but their content could be updated (see section 4.2) if the scenario realization is infeasible (see section 5.1) or if experts can provide reasons for low correlation according to section 5.3. Such updates mainly address the required fidelity of models. For instance, a given vehicle model in combination with a given trajectory following model both separately fulfilling the fidelity requirements of test case but in combination leading to unexpected behaviour (scenario not realized) or low correlation. In such a case, the fidelity requirements of at least one of the models must be increased to avoid such a combination of models in the future.

4.1.2 Test Instance Capabilities

Like test case requirements, test instance capabilities are defined in Deliverable D3.3 [1] and retain their structure. Their content may require updates (Section 4.2) during re-allocation if the scenario cannot be executed (Section 5.1) or experts identify causes of low correlation (Section 5.3). Mainly such updates increase the granularity of ODD descriptions. For example, if a test instance shall cover in ideal cases a specific range of lateral accelerations or specific physical effects of sensors and the test case requirements are near to instance capability boundary. If the test case requirements were finally not fulfilled it must be investigated further if the test instance capabilities need to be updated or described more detailed in order to avoid the allocation of such test cases.

4.1.3 Additional Suggestion – SUT Requirements

In the re-allocation process compared to the initial allocation process the SUT requirements have been introduced if there are any SUT requirements not related to ODD description and not already covered by the test case requirements. Test cases are typically designed for clusters of systems, not a specific SUT. For example, highway test cases validate safety for highway or traffic jam systems, while junction scenarios target low-speed systems. Similarly, test instances, such as simulation tools or hybrid setups, validate safety for multiple systems.

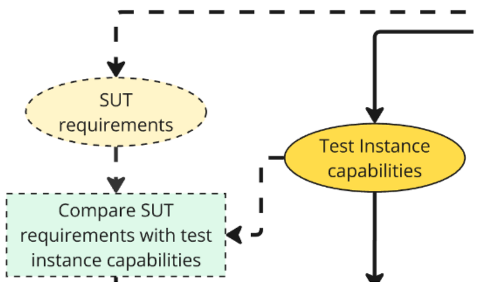

SUT requirements
Test Instance capabilities
Compare SUT requirements with test instance capabilities

Figure 7. SUT requirement additional box, and its relation to the initial allocation scheme

However, the SUT may have specific requirements, particularly implementation-related, not addressed by the test case. For example, a model-based design approach might be chosen to create the system under test and the test instance could be a real-time ECU. In such a case other test instances should be preferred for the allocation that use also the model-based design approach.

Additionally, the SUT may have specific hardware requirements on the test instance. For instance, an artificial intelligence (AI) based SUT might have specific requirements on high-performance GPUs to be executable in real-time. Here, test instances using normal ECUs or are implemented on office laptops should not be allocated. However, most of the SUTs or SUT requirements should be covered by the comparison of test case requirements and test instance capabilities.

If there are any additional SUT requirements not covered by test case requirements they need to be considered in the test instance allocation process. This means to compare these requirements with the test instance capabilities in a similar way the test criteria from the test case requirements are compared to the test instance capabilities.

4.2 Capability and Requirement Refinement from Re-allocation Process

If the intended scenario is not realised or experts identify low correlation, the relevant test case requirements or test instance capabilities must be updated to prevent incompatible allocations in the future. These updates typically adjust ODD descriptions, such as ranges or thresholds, or increase model fidelity requirements (which might also result in updates of the SUT requirements if not covered by the test case requirements). Refinements are often needed because initial capabilities or requirements were estimated or did not account for specific model interactions. After updating the test case requirements, test instance capabilities, or both, repeat the initial allocation process to generate an updated list of feasible test instances for the test case.

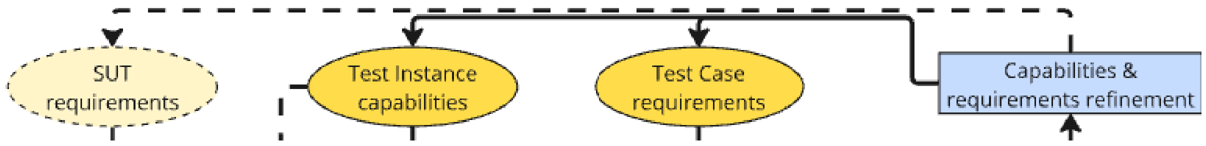

SUT requirements
Test Instance capabilities
Test Case requirements
Capabilities & requirements refinement

Figure 8. Capability and requirements refinement block, and its relation to the initial allocation

5 RE-ALLOCATION WORKFLOW

Test run validation metrics are needed for two reasons: a) in order to verify that the test execution is valid and b) in order to evaluate the test scenario importance and prioritize highly critical scenarios. In the following two sections we deal with both.

5.1 Scenario Realisation

To support the robustness of the SAF and the accuracy of the outcome, it is essential to verify that each scenario is not only executed in the technical sense, but also realised meaningfully at the semantic level. This means checking whether the SUT actually encountered the intended triggering conditions and whether it responded appropriately and in alignment with the scenario's defined logic. A scenario should not be considered validly executed if any of the scenario behaviour phases that test the intended function were never reached.

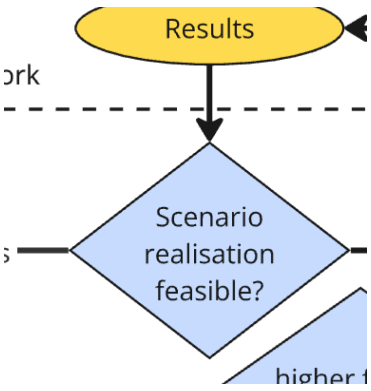


Figure 9. Scenario realisation checking block, and its position in the overall workflow

Scenario realisation involves more than verifying system outputs, it requires assessing whether the test conditions led to the specific stimuli the scenario was designed to probe. For example, in a scenario meant to evaluate the SUT's response to an overtaking and cut-in event, it is not enough to observe that the simulation ran or that the SUT avoided a crash. The scenario is only realised if the overtaking manoeuvre occurred as defined, and the SUT was exposed to the resulting safety relevant situation.

Figure 10 illustrates such a scenario. The yellow vehicle represents the SUT. The blue vehicle is an agent vehicle, and the scenario defines its role as performing an overtaking manoeuvre on the left before cutting back in ahead of the SUT, thereby testing whether the SUT detects the cut-in and reacts safely, typically by slowing down or adjusting its time headway. However, if the blue vehicle fails to complete the manoeuvre (e.g., due to its speed being too low, misaligned initial conditions, or other factors), then the SUT is never actually exposed to the critical situation. Even if the simulation reaches completion without errors or violations, the scenario's objective has not been fulfilled.

This distinction is important. A test case might meet general safety metrics (e.g. no collisions or rule violations), but if the specific behavioural trigger is never reached, the scenario has not been realised in the intended sense. No valid conclusion can be drawn about the SUT's behaviour in that scenario context.

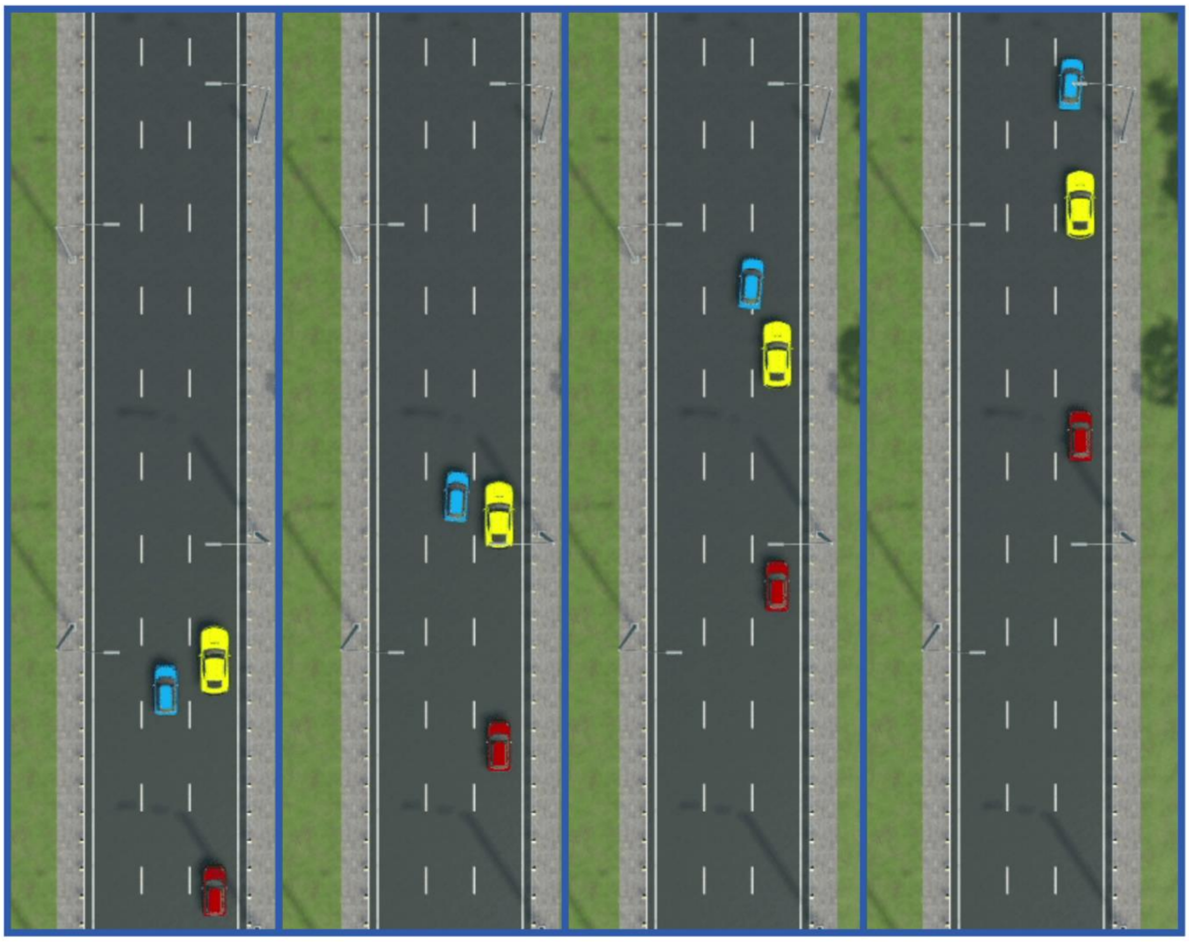


Figure 10. Example scenario execution time sequence snapshots

Such situation can be caused for two reasons: 1) the scenario is wrongly defined, 2) wrong simulation component is used. First, the scenario itself may be wrongly defined, with unrealistic timing, conflicting vehicle dynamics, or ambiguous logic. Such issues are typically detected during quality assurance when scenario files are reviewed or integrated into the scenario database. Second, the test allocation process may produce a test instance that does not align well with the scenario's requirements. For example, the initial allocation might pair the SUT with agent behaviours or speed profiles that make the overtaking manoeuvre impossible, or at least improbable within the defined test window.

Even when the scenario file is correct, allocation mismatches (e.g., incompatible vehicle capabilities or environmental constraints) can prevent the triggering condition from occurring. This can also arise due to simulation tool limitations, where the behaviour of non SUT agents is insufficiently controlled or lacks the fidelity to execute tightly constrained actions like timely lane changes.

In these cases, scenario realisation must be formally flagged as “not achieved.” The cause should then be diagnosed: whether it lies in the scenario logic, the test allocation, or the simulation platform. Appropriate refinements can include adjusting the scenario parameters, tightening the constraints on test instance selection, or improving the agent behaviour control. These refinements may also feed into updates of ODD boundaries, SUT capability descriptions, or scenario metadata, ensuring that future allocations increase the likelihood of realising the intended behaviours.

When a scenario intention is not realised, the causes of each of the two reasons can be further traced. Reason #1 (wrong scenario logic) is dealt with at the quality assurance stage when the individual SCDBs are connected to the SUNRISE Data Framework. However, reason #2 (mismatching/wrong simulation component) can indicate the initial allocation might carry incorrections. The initial allocation is primary based on the cross-matching between identified requirements from the scenario, and the known capabilities of the test instances.

As consequence, if the scenario did not happen as indicated, the (potential) reasons need to be provided to the refinement of capabilities and requirements (section 4.2).

5.2 Test Run Validation Metrics and Higher Fidelity Test Instance Execution

In this section, two main categories of **test run validation metrics** are presented, one of which is based on **safety confidence**, the other is based on **correlation between higher and lower test instances**. Both categories are explained in detail below. In addition to the validation metrics, the process and consideration of high fidelity test instance execution is expanded, which sits in between the safety confidence evaluation and correlation analysis.

Please note that the test run validation metrics differ from those used in the ‘Test Evaluate’ block. The test run validation metrics are used to validate whether the correct test instance is used, and consequently validation of the test run, regardless if the SUT passes or fails the test. Whereas the metrics used in ‘Test Evaluate’ block in the SAF are used to assess the performance of the SUT, therefore result into the pass or fail of the SUT. The test run validation metrics sit within the ‘Allocate’ block of the SAF, with the feedback loop carrying the information of the current test run from the Safety Argument box back to the Environment box.

In the following some general remarks are given, before considering the two main categories in detail.

When transitioning from a lower fidelity test instance to a higher fidelity test instance, for example on a proving ground, the key challenge lies in selecting a small but representative set of test scenarios. These scenarios should validate the reliability of the simulation outcomes while maximizing the insight gained from each physical test. Given the constraints of time, cost, and feasibility inherent to proving ground testing, a systematic and data-driven selection process is essential.

- The first step involves analysing the results of the simulation study to understand the structure and behaviour of the explored scenario space. This includes **identifying**

clusters of similar outcomes, **critical transitions** in behaviour (such as from safe to unsafe), and regions of **high sensitivity** where small changes in parameters lead to large variations in outcomes. Clustering algorithms such as k-means (which is an unsupervised clustering algorithm that partitions data into *k* groups by minimising the distance between data points and their assigned cluster centroids) can help detect regions in the parameter space where system behaviour is similar. Additionally, sensitivity analysis, for example variance based, can highlight which parameters most influence the outcome, guiding to focus on those dimensions in physical testing.

- Once one has a structured understanding of the scenario space, one can adopt a multi-pronged test selection strategy that balances representativeness, risk, and information gain. One key approach is **importance sampling**, where a few scenarios from each identified cluster are selected to ensure that all typical behaviour patterns are covered by the physical tests. This prevents blind spots in the validation and gives confidence that the simulation results generalize well across different parts of the space.

- Another important strategy is **boundary testing**. In many safety-critical systems, outcomes often hinge on specific conditions, such as a car barely avoiding a collision or a system reacting just in time. These conditions form the “decision boundaries” in the scenario space. Selecting points near these transitions helps validate whether the simulated boundaries accurately reflect real- world behaviour and ensures that safety confidences are not overstated.

- In addition to representative and boundary points, it is crucial to **test high-risk scenarios**. These are situations where the simulation indicated failure, near-failure, or extreme behaviour. Validating these ensures the credibility of the simulation in critical use cases and supports safety assurance.

- If a **surrogate model** was developed, such as a Gaussian process, decision tree, or neural network, to approximate the simulation results, one can leverage it to further guide test selection. Such models allow for uncertainty-based sampling, where scenarios with high model uncertainty are targeted. This is especially useful if one is using active learning or Bayesian optimization techniques, as it focuses physical testing on scenarios that are both poorly understood and potentially of high impact.

- Finally, **principles from the Design of Experiments (DoE)** literature can be incorporated to refine the test selection. Techniques such as Latin Hypercube Sampling (LHS) provide good space-filling properties, while maximin designs ensure selected points are spread out as much as possible in the parameter space, reducing redundancy. Adaptive DoE methods allow for iterative refinement, where each round of physical testing informs the next.

Throughout this process, practical constraints must be considered. The selected scenarios should be feasible to reproduce on the proving ground, both technically and logistically. Collaborating with test engineers early in the selection process can help filter out impractical test cases and suggest adjustments to scenario definitions. Also, given the cost of physical testing, scenarios that offer the highest information-to-cost ratio should be prioritized.

5.2.1 Safety Confidence

An important consideration is the safety confidence that is accepted. The safety confidence is defined to be the quantified margin by which a system must exceed minimum performance thresholds to account for model uncertainty and false acceptance risks. False acceptance is denoting the misclassification of a scenario as safe when it is not. Any test that is conducted is, by its design, only an approximation of the real-world situations a CCAM vehicle will face on the road. This introduces a degree of uncertainty in the results of these tests. To overcome this, there are several possible solutions. A large number of tests will reduce the uncertainty, as will the validation of the results in a higher test instance. One additional factor to consider is the introduction of a safety confidence that requires a higher degree of safety than might be needed by the safety KPIs themselves. Such KPIs are dependent on the use cases, therefore inherited from the input layer to the SAF.

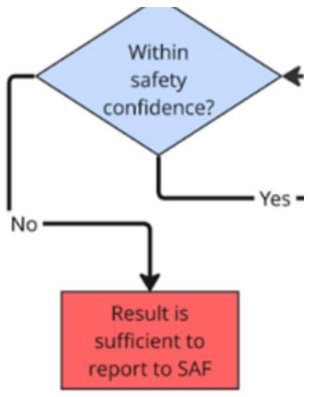

Within safety confidence?
Yes
No
Result is sufficient to report to SAF

Figure 11. Safety confidence checking block, and its position in the overall flow

The safety confidence can be thought of as moving the pass-fail boundary towards the safe region in the parameter space of the logical driving scenario. Around the pass-fail boundary, there are confidence bands denoting the uncertainty in the estimation of the pass-fail boundary. Such uncertainty could be calculated using surrogate models as described in deliverable D3.4 [6]. A detailed explanation on how to derive the probability of system failure from a surrogate model is given in [7] and [8]. The safety confidence can now be understood as moving the pass-fail boundary towards the confidence band that represents the false-acceptance risk that is still acceptable.

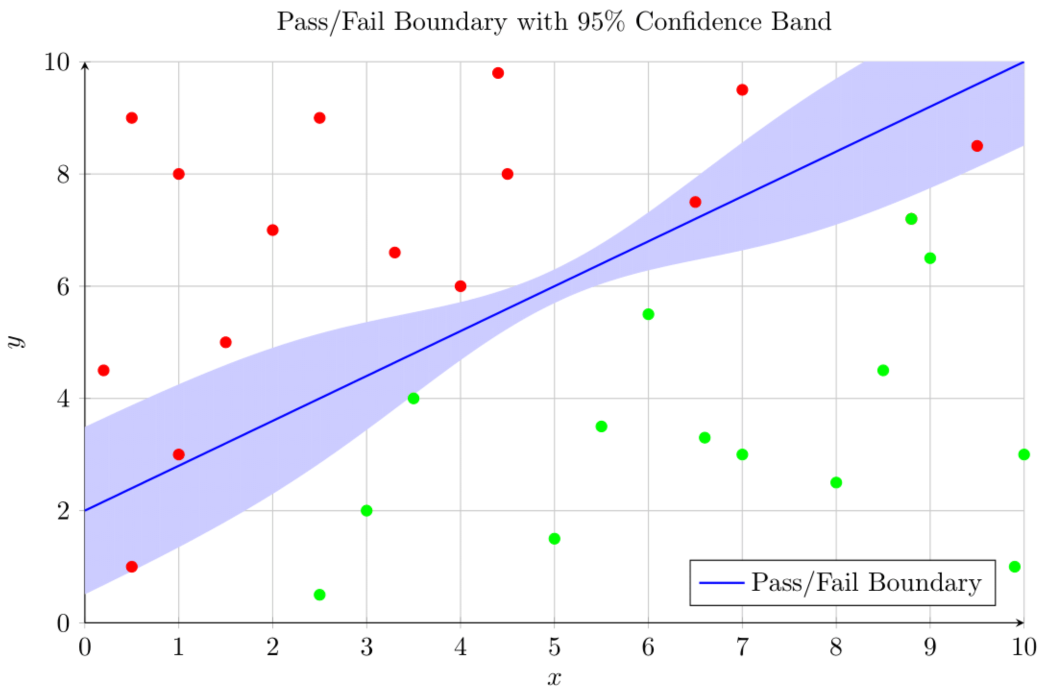


Figure 12. Pass/Fail boundary with confidence band

This concept is illustrated in Figure 12. Here an estimation of the pass-fail boundary in a two-dimensional parameter space is given. Around the pass-fail boundary a confidence band indicates the confidence in this prediction. Red points denote failed tests, and green points denote successful tests. Even though two points lie in a region estimated to pass, since they are within the confidence band they must be considered as failed. In the region of these tests, the confidence in the pass-fail boundary is not high enough.

To use safety confidence to validate the result of one test the following steps should be followed:

1. An acceptable false acceptance risk must be defined.
2. The test must be performed.
3. An estimation of the uncertainty at the test point must be established.
4. The uncertainty must be compared to the false acceptance risk to ensure sufficient confidence.

If by performing these steps it becomes clear that confidence in the test result is too low, the test should be seen as failed. This information, is then used in order to take decisions on specific set of scenarios to be allocated in higher-fidelity testing environments, i.e. hybrid testing, as described in the next section. The derivation of the confidence interval of the pass-fail boundary is highly related to the methodology developed in deliverable D3.4 [6] for parameter space exploration and parameter space coverage estimation. Please note that as described in deliverable D3.4 [6], when using surrogate models to estimate the pass/fail boundary, a probabilistic interpretation of the scenario space pass/fail outcome is obtained without the need to execute all the concrete test scenarios.

An alternative method to derive safety confidence is to test a single concrete scenario multiple times. This has been used in the testing of automated driving systems by [9], which the confidence level in a single concrete scenario is estimated using a multinomial distribution modelling the results of multiple executions of the same scenario.

5.2.2 Execute at Higher Fidelity Instance

In this part of the process the decision is made if it is possible to apply a higher fidelity test instance. In the case no other test instance is available that would improve the fidelity, all results so far as well as the allocation limitation need to be reported to the SAF. However, if it is feasible to test in a higher fidelity test instance, then the identical conditions (e.g., scenario parameters, test evaluate metrics etc.) as used in the lower fidelity test instance run should be used in the higher fidelity test instance run.

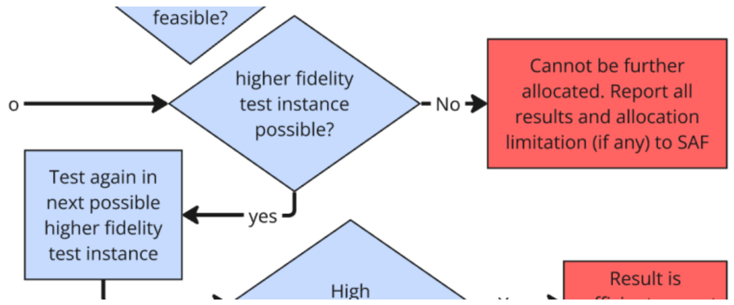

feasible?
higher fidelity test instance possible?
No
Cannot be further allocated. Report all results and allocation limitation (if any) to SAF
Test again in next possible higher fidelity test instance
yes
High
Result is

Figure 13. Higher instance execution block, and its position in the overall flow

To enhance the fidelity of a test instance, several strategies can be employed.

- Firstly, higher-accuracy models for **vehicle dynamics** can be integrated, incorporating more detailed representations of suspension systems, tire-road interactions, and powertrain behaviour.
- Similarly, **sensor simulation** can be improved by adopting physics-based sensor models that account for phenomena such as noise, occlusion, reflection, and weather effects, thus better approximating real-world sensor behaviour.
- The **environmental model** can also be refined by increasing the level of detail in static and dynamic elements, such as road geometries, lane markings, traffic signs, and surrounding infrastructure.
- Additionally, **traffic behaviour models** can be replaced with more realistic agent-based approaches, enabling more nuanced interactions between road users.
- Another important aspect is the synchronisation and interoperability between these subsystems. Using a **co-simulation framework** or **standardised interfaces** (e.g., ASAM OSI, FMI) can facilitate integration without compromising accuracy (see deliverable D4.4).

- Finally, **validation against real-world data** through Hardware-in-the-Loop (HiL) or Vehicle-in-the-Loop (ViL) setups can ensure that increasing fidelity translates into meaningful improvements in test quality and predictive power.

A very important consideration to be made is the determination of points to be tested in the higher fidelity test instance. The SUNRISE test instance allocation process developed in task 3.4 (see deliverable D3.3 for more details) proposes the use of most-efficient (e.g., test throughput) test instances first, that of course still have to fulfil all the other necessary requirements determined by the allocation process. These more efficient test instances allow for testing of a greater number of concrete scenarios and thereby more coverage of the scenario parameter space with the tests. This however means that if the results need to be validated in a higher fidelity test instance, the capability to test the same number of concrete scenarios is not sufficient. Instead, one has to choose a smaller number of test cases in such a way as to cover the parameter space well. The results of the tests in the higher test instance can then be correlated with those in the lower test instance to determine if those results can be accepted.

A straightforward way to choose sample points to be tested in the higher fidelity test instance is the use of a scoring system to be applied on a number of candidate points in the parameter space. First, a set of candidate points has to be defined. A simple way is to distribute the candidate points equidistantly in the parameter space. The number of candidate points should be chosen on the order of a magnitude higher than the number of tests that should be conducted. For each candidate point a score can be calculated based on the proximity of the candidate point to the original sample point and the variance at the candidate point. The different scores can then be weighted and aggregated. The candidate points with the highest scores should be chosen to conduct the validation in the higher fidelity test instance. An additional step that can be performed is to apply a minimum distance between candidate points to improve coverage of the parameter space.

5.2.3 Examine Correlation

To evaluate how well simulation results reflect real-world outcomes, one commonly used method is the Pearson correlation coefficient. This statistical measure quantifies the strength and direction of a linear relationship between two continuous variables, in this case, the evaluation metric obtained from simulation and the corresponding metric from real-world testing, such as time to collision (TTC). An overview of this type of correlation coefficient can also be found in [10].

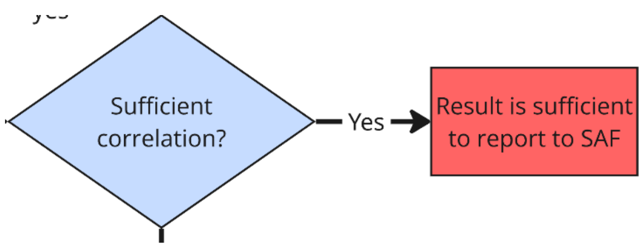


Figure 14. correlation comparison block, and its position in the overall flow

To apply the Pearson correlation, firstly the paired data points from both sources under identical conditions is gathered. For example, for each test case (like a specific initial speed). The TTC from both the simulation and the proving ground test can be used for such a comparison. The Pearson correlation coefficient (denoted as r) can then be calculated, which ranges from -1 to +1. A value of +1 indicates a perfect positive linear correlation (the two datasets increase together), 0 indicates no linear relationship, and -1 indicates a perfect negative linear correlation.

The formula behind r considers the covariance between the two datasets, normalized by their standard deviations. In practical terms, this can be easily computed using statistical software. If the resulting r value is close to 1 and statistically significant (usually indicated by a low p-value), it suggests that the simulation and real-world measurements are closely aligned in a linear sense, thereby providing confidence in the simulation model's ability to replicate real-world behaviour. It is important to define an acceptable level of correlation before conducting the correlation tests.

However, it is important to remember that Pearson's correlation only captures linear relationships. If the relationship is nonlinear or contains systematic bias (e.g., consistent over- or underestimation), this method alone may not be sufficient, and additional analyses may be appropriate.

5.3 Experts Explanation

There are many potential explanations to achieving lower correlation than expected. The expert needs to evaluate this and motivate if a re-allocation is expected to result in better and sufficient correlation, or if insufficient correlation is expected also after relocation and thereby necessary to further investigate.

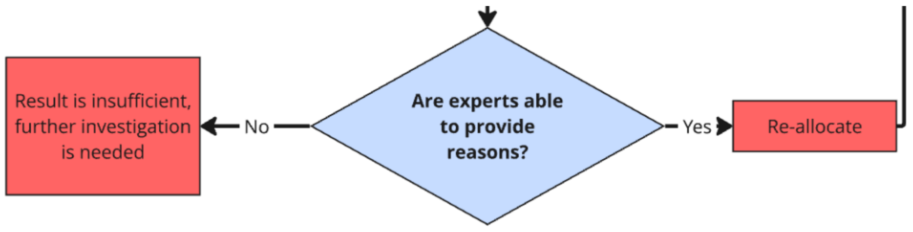

Result is insufficient, further investigation is needed
No
Are experts able to provide reasons?
Yes
Re-allocate

Figure 15. Expert explanation block, and its position in the overall flow

Following the SAF structure as shown in Figure 3, a walkthrough of the different SAF blocks proposes several possible explanations to the achieved low correlation:

1) The first relevant SAF block to analyse is the Input block, shown in the bottom of Figure 3. The input is essential, and insufficient specifications may result in unintended results from one or all test instances and thereby low correlation. Possible insufficiencies include among other:
    - The ODD description is insufficient, e.g., there may be conditions that have significant impact on the proving ground but are not considered in simulations as they are not in the ODD description.

- The behaviour description is not sufficient resulting in the SUT showing different behaviour in different test instances.
- The test objectives are not correctly specified.
- The test instance capabilities are wrongly described.

2) The second relevant SAF block is the Scenario block shown on the left side of Figure 3. As the Scenario block mostly is outside SUNRISE scope, this text focuses on the impact on the Scenario block to downstream blocks rather than the Scenario block itself. An important aspect is:
   - The fetched scenarios do not match the queries.

3) The third block to analyse is the Environment block in the middle of Figure 3. Stepping through the sub-blocks at least following possible explanations to low correlations between test instances may be identified:

   Query and Concretize
   - The defined queries do not match the input specifications.
   - The concretized scenarios do not match the SUT or ODD requirements.

   Allocate
   - The test instance capabilities are not correctly specified.
   - The test cases have not been correctly allocated to test instances.

   Execute
   - The scenario has not been executed as expected in one of test instances, e.g., as exemplified in Section 5.1, or due to too low fidelity in (parts) of the simulation.
   - The test scenario cannot be correctly executed in one or several of the test instances.

For the Execute block, a deeper dive can be done for virtual simulation looking into the scheme shown in Figure 16. Examples of possible reasons for low correlation includes:

- Sensor models weakness
- AD function in simulator differs from real vehicle
- Weaknesses in the vehicle dynamics modelling
- Weaknesses in the environment simulation

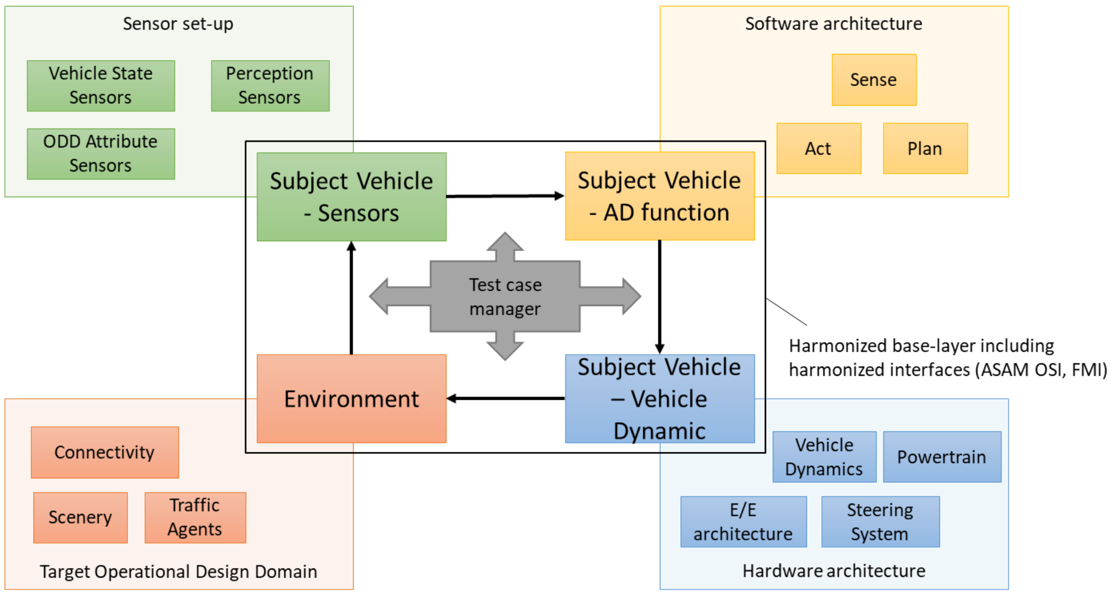


Figure 16. SUNRISE harmonized V&V simulation framework.

Based on analysis of the low correlation results, the expert should conclude how to proceed:

1) A re-allocation to another test instance, e.g., a higher fidelity test instance is expected to result in sufficient correlation.

2) The low correlation cannot be sufficiently explained and need to be further investigated.

6 CONCLUSIONS

This deliverable presents a systematic approach to validating test runs and dynamic re-allocation of test cases to test instances within the SUNRISE SAF. The methodology builds on deliverable **D3.3** [1], which enables scenarios to be mapped to the appropriate test instances in a way that reflects both **system capabilities** and **scenario requirements**. This deliverable D3.5 fits within the wider SUNRISE SAF, specifically within the **'Allocate'** block inside the **'Environment'** block, receiving test execution outcome from the **'Safety Argument'** block. By doing so, it enhances the trustworthiness, relevance and credibility of test outcomes used for safety assessments.

The test run validation includes checking of **scenario realisation** (section 225.1), checking **safety confidence** (section 5.2.1), and **examining the correlation** (section 5.2.3) between higher and lower test instances. The scenario realisation checks whether the intended trigger conditions and behavioural patterns defined in a scenario are actually encountered during testing, if not then subsequent test outcome will not be relevant.

For safety confidence checking and correlation assessment, two core metric types are introduced: **safety confidence**, which quantifies certainty in test outcomes by accounting for model uncertainty; and **correlation level**, which measures the alignment between low- and high-fidelity test results. To select a small, representative set of high-fidelity tests, the approach (section 5.2) combines **clustering**, **sensitivity analysis**, **importance sampling**, and **surrogate modelling**. Scenario selection aims to balance representativeness, risk, and information gain while also accounting for feasibility and cost. Safety confidence is used to shift pass/fail thresholds conservatively to **mitigate false acceptance**. Where feasible, high-fidelity testing uses more accurate models, sensor simulations, and co-simulation frameworks to enhance realism. The Pearson correlation coefficient is discussed to assess alignment between simulation and physical test results, though non-linear effects may require additional analysis. Based on the validation of the test run, the outcome can be: 1) the **test outcome is valid, relevant, and trustworthy,** and the progress of the safety assurance activities can continue to safety argument stage, 2) **re-allocate the test case to higher fidelity test instance and re-run the test**, at the same time update the knowledge on test case requirement and test instance capabilities.

This deliverable plays an important and essential role within the SAF workflow, **ensuring the validity and trustworthiness** of test run results. Without this mechanism, the entire Safety Argument block of the SAF could be built on invalid test outcomes. Furthermore, once such invalidity is identified, this deliverable also deals with the **remedy approaches**, such as execute the test case in a higher fidelity test instance, and update the knowledge on the capabilities of the available test instances. Playing an essential role in various SAF blocks (Allocate, Execute and Test Evaluate), this deliverable is relevant to all SAF users. For industry entities (like test engineers), it lays out the mechanism to ensure test outcomes are trustworthy. For certifiers and regulators, it highlights the evidences that need to be considered when assessing whether safety evidence is trustworthy. For tool developers (e.g., virtual test tools), it provides a mechanism to validate the test instances capabilities against the intended test requirements that the tools should fulfil. Future work includes to stress test the methods for test run validation and dynamic allocation across a wider set of use cases, identify improvement points and refinements.

7 REFERENCES

[1] B. Hillbrand, M. Kirchengast, A. Balis, I. Panagiotopoulos, T. Menzel, T. Amler, E. Collado, J. Beckmann, M. Skoglund, A. Thorsén, T. Singh, M. Shabbir Ali and M. Nieto, “D3.3 Report on the Initial Allocation of Scenarios to Test Instances,” 2024.

[2] “StreetWise: scenario based safety assessment for automated driving,” [Online]. Available: https://www.tno.nl/en/digital/smart-traffic-transport/safety-assessment-automated-driving/streetwise/. [Accessed 23 7 2025].

[3] “Safety PoolTM scenario database,” [Online]. Available: https://www.safetypool.ai/. [Accessed 23 7 2025].

[4] e. ASAM, “OpenSCENARIO V1.2.0,” 2022. [Online]. Available: https://www.asam.net/standards/detail/openscenario/.

[5] “BSI Flex 1889 - Natural language description for abstract scenarios for automated driving systems - Specification,” BSI, 2022.

[6] A. Thorsén, E. de Gelder, M. Kirchengast , B. Hillbrand , T. Menzel , T. Amler , F. Hadj Selem, P. Irvine , E. Daskalaki , E. Kaynar , J. M. TORRES CAMARA, M. Nieto , E. Arnoux and J. Beckmann, “D3.4 Report on Subspace Creation Methodology”.

[7] A.-G. Sorokin and V. Rao, *Credible Intervals for Probability of Failure with Gaussian Processes,* arXiv preprint arXiv:2311.07733, 2023.

[8] J. Li and H. Wang, *Gaussian Processes Regression for Uncertainty Quantification: An Introductory Tutorial,* arXiv preprint arXiv:2502.03090, 2025.

[9] F. Reisgys, C. Steinhauser, A. Schwarzhaupt and E. Sax, “How Uncertainty Affects Test Results for Driving Automation,” in *IEEE International Automated Vehicle Validation Conference (IAVVC)*, 2023.

[10] A.-G. Asuero, A. Sayago and A.-G. González, “The correlation coefficient: An overview,” *Critical reviews in analytical chemistry,* vol. 36, no. 1, pp. 41-59, 2006.

[11] E. de Gelder, J.-P. Paardekooper, A. Khabbaz Saberi, H. Elrofai, O. Op den Camp, S. Kraines, J. Ploeg and B. De Schutter, “Towards an Ontology for Scenario Definition for the Assessment of Automated Vehicles: An Object-Oriented Framework,” *IEEE Transactions on Intelligent Vehicles,* vol. 7, pp. 300-314, 2022.

[12] C. Chang, D. Cao, L. Chen, K. Su, K. Su, Y. Su, F.-Y. Wang, J. Wang, P. Wang, J. Wei, G. Wu, X. Wu, H. Xu, N. Zheng and L. Li, “MetaScenario: A Framework for Driving Scenario Data Description, Storage and Indexing,” *IEEE Transactions on Intelligent Vehicles,* vol. 8, pp. 1156-1175, 2022.

[13] E. de Gelder, H. Elrofai, A. Khabbaz Saberi, O. Op den Camp, J.-P. Paardekooper and B. De Schutter, “Risk Quantification for Automated Driving Systems in Real-World Driving Scenarios,” *IEEE Access,* vol. 9, pp. 168953-168970, 2021.

[14] ISO 34501, *Road Vehicles - Test Scenarios for Automated Driving Systems - Vocabulary,* International Organization for Standardization, 2022.

[15] E. de Gelder, J. Hof, E. Cator, J.-P. Paardekooper, O. Op den Camp, J. Ploeg and B. De Schutter, “Scenario Parameter Generation Method and Scenario Representativeness Metric for Scenario-Based Assessment of Automated Vehicles,” *IEEE Transactions on Intelligent Transportation Systems,* vol. 23, pp. 18794--18807, 2022.

[16] ECE/TRANS/WP.29/2021/61, “New Assessment/Test Method for Automated Driving (NATM) - Master Document,” World Forum for Harmonization of Vehicle Regulations, 2021.

[17] H. Weber, J. Bock, J. Klimke, C. Roesener, J. Hiller, R. Krajewski, A. Zlocki and L. Eckstein, “A Framework for Definition of Logical Scenarios for Safety Assurance of Automated Driving,” *Traffic Injury Prevention,* vol. 20, pp. S65--S70, 2019.

[18] ISO 34503, *Road Vehicles - Test Scenarios for Automated Driving Systems - Taxonomy for Operational Design Domain for Automated Driving Systems,* International Organization for Standardization, 2023.

[19] ISO 34504, *Road Vehicles - Test Scenarios for Automated Driving Systems - Scenario categorization,* International Organization for Standardization, 2024.

[20] X. Zhang, S. Khastgir, H. Asgari and P. Jennings, “Test Framework for Automatic Test Case Generation and Execution Aimed at Developing Trustworthy AVs from Both Verifiability and Certifiability Aspects,” in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2021.