SUNRISE

SAFETY ASSURANCE FRAMEWORK FOR CONNECTED, AUTOMATED MOBILITY SYSTEMS

D3.3

Report on the Initial Allocation of Scenarios to Test Instances

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	D3.3: Report on the Initial Allocation of Scenarios to Test Instances
Deliverable responsible	Bernhard Hillbrand, Virtual Vehicle Research
Status - Version	Final – V1.0
Submission date	29/08/2024
Keywords	CCAM, Safety Assurance, Scenarios, Requirements, Allocation, Test Cases, Test Instances

Authors/Contributors

Name	Organisation
Bernhard Hillbrand, Martin Kirchengast	VIF
Athanasios Balis, Ilias Panagiotopoulos	ICCS
Thaddaeus Menzel, Thomas Amler	IDI DE
Eloy Collado	IDI
Jobst Beckmann	ika
Martin Skoglund, Anders Thorsén	RISE
Tajinder Singh	SISW
Mohammed Shabbir Ali	VED
Marcos Nieto	VICOM

Quality Control

	Name	Organisation	Date
Peer review 1	Darko Stern	AVL	10/07/2024
Peer review 2	Jobst Beckmann	ika	10/07/2024
Peer review 3	Stefan De Vries	IDI	10/07/2024

Version history

Version	Date	Author	Summary of changes
0.1	14/02/2024	Bernhard Hillbrand	First draft of document structure

0.2	01/07/2024	All authors	Creating a 1st draft version
0.3	11/07/2024	All authors	Creating a 2nd draft version according to the feedback of the reviewers.
0.4	15/08/2024	All authors	Finished a 2nd draft version according to the feedback of the reviewers
0.5	28/08/2024	All authors	Finished a 3rd draft version according to the feedback of the reviewers
0.6	29/08/2024	Bernhard Hillbrand	Changed the 3rd draft version according to the feedback of the coordinator
1.0	29/08/2024	Bernhard Hillbrand	Final version ready for submission

Legal disclaimer

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Climate, Infrastructure and Environment Executive Agency (CINEA). Neither the European Union nor the granting authority can be held responsible for them.
Copyright © SUNRISE Consortium, 2024.

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
B2X	Bicyclist-to-Everything communication
BSI	British Standards Institution
CCAM	Connected, Cooperative, and Automated Mobility
COTSATO	COncretizing Test Scenarios and Associating Test Objectives
CPM	Collective Perception Message
CV2X	Cellular-vehicle-to-everything
DiL	Driver-in-the-Loop
DNN	Deep Neural Network
ETSI	European Telecommunications Standards Institute
Euro-NCAP	European New Car Assessment Programme
FOT	Field Operational Test
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HiL	Hardware-in-the-Loop
HMI	Human Machine Interface
I2X	Infrastructure-to-Everything communication
ISMR	In-Service Monitoring and Reporting
ISO	International Organization for Standardization
ITS	Intelligent Transportation System
MBS	Multi-Body-System

MuCCA	Multi-Car Collision Avoidance
NATM	New Assessment/Test Method for Automated Driving
ODD	Operational Design Domain
OEM	Original Equipment Manufacturer
PDF	Probability Density Function
SAE	Society of Automotive Engineers
SAF	Safety Assurance Framework
SCDB	SCenario DataBase
SUNRISE	Safety assUraNce fRamework for connected, automated mobIlity SystEms
SUT	System Under Test
UC	Use Case
URI	Uniform Resource Identifier
V&V	Verification and Validation
V2X	Vehicle-to-Everything communication
VUT	Vehicle Under Test
XiL	X-in-the-Loop

EXECUTIVE SUMMARY

The safety assurance of Connected, Cooperative, and Automated Mobility (CCAM) systems is crucial for their successful adoption. The **S**afety ass**U**ra**N**ce f**R**amework for connected, automated mob**I**lity **S**yst**E**ms (SUNRISE) project develops a Safety Assurance Framework (SAF) that enables the safety assurance of CCAM systems. Due to the infeasibility and impracticality of assuring safety solely through test drives, scenario-based testing forms a substantial part of the SAF.

This document describes the **process of the initial allocation of test cases to test instances**. This process is divided into **two parts**. In the first part, the requirements of the test case are compared with the capabilities of the available test instances to see where the test cases could be executed. This determines if a certain test instance will be able to execute the specific test case. In the second part of the process, the decision-making with respect to certain metrics is done.

For the **first part**, a structure has been defined that is partly based on the ISO34503 standard [1] to guide the user of the SUNRISE safety assurance framework through this process of comparing the test case requirements with the test instance capabilities. The structure is divided into four main parts (see chapter 3):

- **Scenery Elements**: this describes the physical environment of the test case, including details like road types, junctions, and special structures (tunnels, bridges). It can also include temporary changes like road works.
- **Environmental Conditions**: this captures weather conditions (temperature, wind, rain, snow), connectivity (communication and positioning), particulates (fog, dust), and illumination (day, night) that can affect sensor performance and overall system behaviour.
- **Dynamic Elements**: this describes the moving objects in the scenario, including the subject vehicle and other traffic agents (vehicles, pedestrians, cyclists, etc.). It also details the behaviour models used to represent these agents in simulations.
- **Test Criteria**: this describes the additional criteria that are important but not part of a scenario description like evaluation criteria, completion criteria and capability criteria.

The **second part** of the document describes the decision-making process for allocating test cases to different testing environments (see chapter 4). The process is based on various metrics that include factors like criticality, sensor fidelity, and cost. Safety is the top priority since this process is part of the Safety Assurance Framework of SUNRISE. The process prioritises virtual simulations ("virtual simulation first" approach) due to their lower cost and effort. However, it also considers the limitations of simulations and may require further testing in higher-fidelity environments (hardware-in-the-loop or proving grounds) in specific scenarios.

The document outlines a step-by-step decision process for allocating test cases. This process involves:

- **Evaluating the need for further testing** in higher-fidelity environments based on factors like SUT behaviour meeting safety criteria with a small margin, the test case lying in a region with limitations in the current testing environment or the test case involving potential triggering conditions for safety issues.
- **Iteratively allocating tests** to the next cheapest higher-fidelity environment until sufficient results are obtained.
- **Documentation**: The results of the allocation process are documented and reported to the safety assessment framework (SAF) for further analysis.

This approach ensures that self-driving vehicles are thoroughly tested in a cost-effective manner. By prioritising virtual simulations and strategically allocating tests to higher-fidelity environments based on safety-critical factors, this process optimises the testing process while meeting safety requirements.

While this document provides useful and structured information related to the allocation of test cases to test instances, it should not be observed as a full guide or instructions manual. For the comparison of test case requirements and test instance capabilities for example, it is not feasible to provide all details of an ODD structure, due to the wide variety of test instances and an even wider variety of test cases. Applying the contents of this deliverable may therefore require additional efforts or expertise to successfully allocate test cases to test instances.

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 the following Figure 1 indicates, it takes a central role, fulfilling the needs of different automotive stakeholders that all have their own interests in using it.

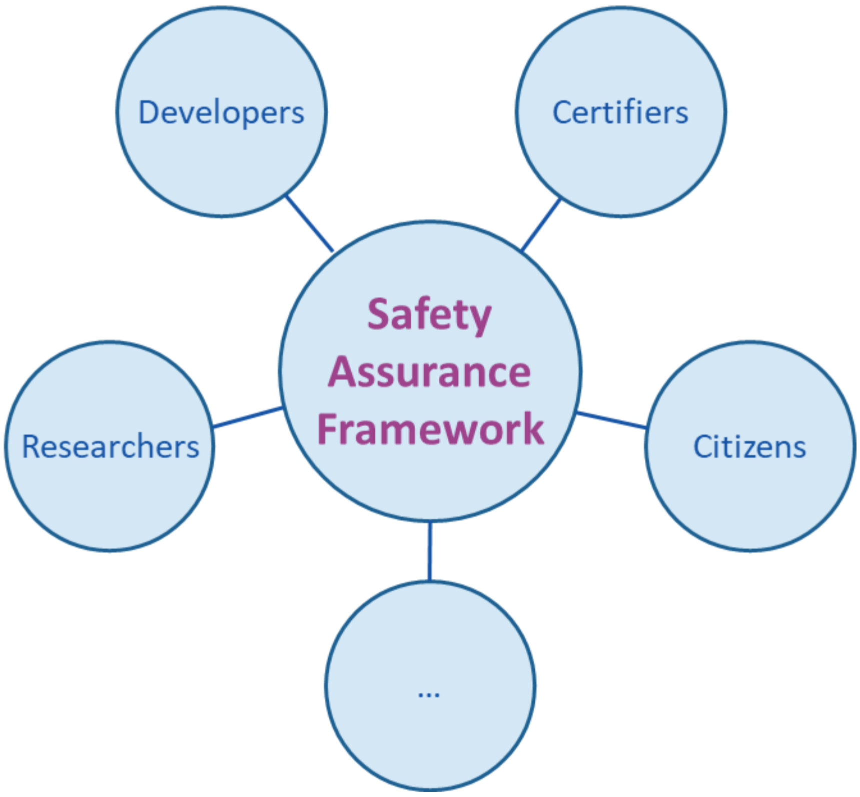


Figure 1: Safety Assurance Framework stakeholders

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.

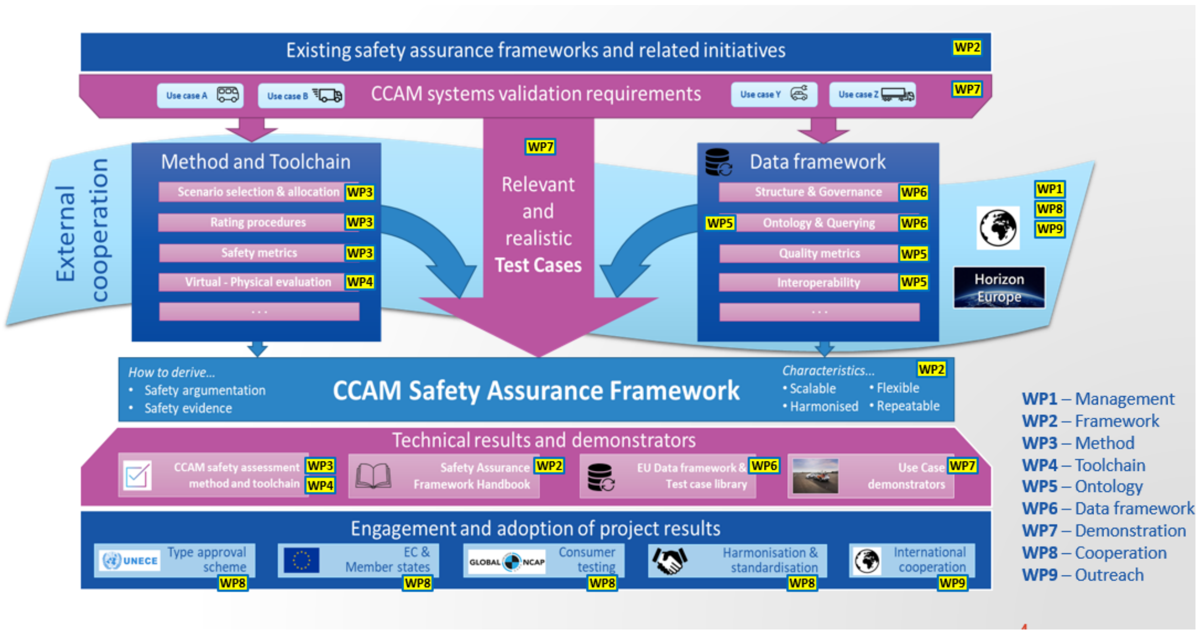

Existing safety assurance frameworks and related initiatives
WP2
Use case A
Use case B
CCAM systems validation requirements
Use case Y
Use case Z
WP7
External cooperation
Method and Toolchain
Scenario selection & allocation WP3
Rating procedures WP3
Safety metrics WP3
Virtual - Physical evaluation WP4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

1.2 Purpose of the deliverable

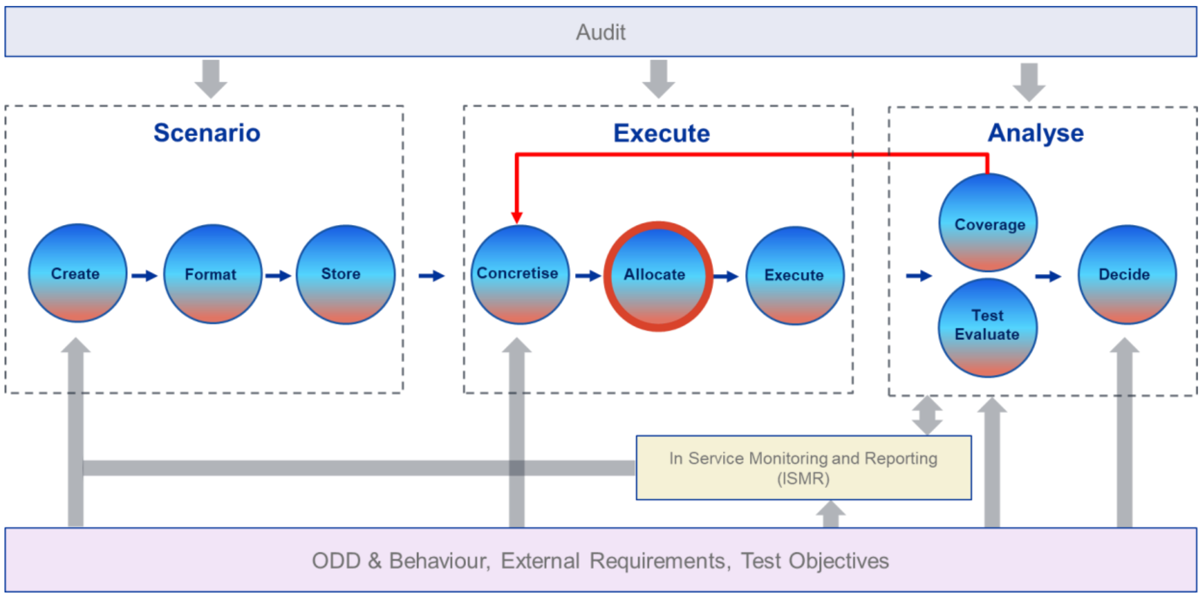
This deliverable is part of work package 3 “CCAM V&V Methodology for Safety Assurance”, which aims to define and condense an overall methodology to support the safety argumentation. One aspect of this work is task 3.4 “Initial allocation of Scenarios to Test instances”. This explains the approach for assigning test cases to appropriate test instances during initial allocation. The objective is to balance test execution efficiency with the reliability of the obtained results. The decision-making process leverages both functional and non-functional metrics to achieve this balance. The approach can be split into two parts. First, a comparison is performed between the test case requirements and the test instance capabilities (see chapter 3).

The proposed decision-making process for the initial allocation of test cases to test instances is elaborated upon in chapter 4). Functional and non-functional metrics are considered to evaluate the suitability of test instances for test cases. Additionally, a method is proposed to address the trade-off between test result efficiency and reliability by testing selected test cases in higher fidelity test instances.

This deliverable describes the process and necessary steps to allocate the test cases in a meaningful way. The test case as well as a technical description of the test instances are the input to this process and the *allocated* test cases are the output.

Despite the name of the deliverable being “Initial allocation of Scenarios to Test instances” this deliverable will talk about test cases. The scenario description is only one part of the test case and therefore, other aspects of the test case need to be considered for the initial allocation as well.

Figure 3 shows the location of this work in the overall draft version of the SUNRISE Safety Assurance Framework (SAF).



Audit
Scenario
Execute
Analyse
Create
Format
Store
Concretise
Allocate
Execute
Coverage
Test Evaluate
Decide
In Service Monitoring and Reporting (ISMR)
ODD & Behaviour, External Requirements, Test Objectives

Figure 3: The draft version of the SUNRISE Safety Assurance Framework with its main parts.

1.3 Intended audience

This deliverable serves multiple stakeholders. One stakeholder is work package 7, as this deliverable presents the recommended way to allocate test cases to test instances for the use cases in work package 7. The methodology described in this report provides the necessary steps to derive which test instances would be capable of executing a certain test case. It also provides a decision-making process in case multiple test instances would be capable of executing a test case to decide to which test instance a test case should be allocated first. Also, other technical work packages of SUNRISE can be considered part of the audience. D3.3 provides the foundational mapping of scenarios to appropriate test instances, which then enables the dynamic allocation in task T3.5, tool integration in WP4, and data handling in WP6.

This methodology is also valid for external users of the SUNRISE SAF and provides the same advantages and functionalities to them as well. The process described in this deliverable provides the methodology to allocate the test cases to the available test instances, which is important to assure a flawless execution.

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

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

Chapter 2 gives an overview of the process to initially allocate test cases to test instances. It introduces the different phases of the process.

Chapter 3 explains the process of comparing the requirements from the test cases with the capabilities of the test instances. It describes how these requirements and capabilities can be extracted and which structure is used to compare.

Chapter 4 explains the decision-making process and the steps necessary for the initial allocation of the test cases to a certain test instance in more detail, as well as the metrics that need to be considered for this process.

2 OVERVIEW OF THE INITIAL ALLOCATION PROCESS

The initial allocation process as displayed in Figure 4 has two inputs. The first input is provided by the prior step of the SAF. Test cases have been prepared involving a scenario description and other additional aspects necessary for the execution like the expected behaviour of the system under test (SUT) and pass/fail criteria. The information contained in the individual test cases is extracted to get the requirements for that test case. What this extraction can look like is described in the chapter 3.1. The other inputs for the process are the available test instances and their capabilities. Test instances include virtual testing, any form of X-in-the-Loop (XiL) testing and proving ground testing. Field testing is not considered as an option because of the uncontrollable nature of these tests, that makes it hard to reproduce certain scenarios. Chapter 3.2 describes how the capabilities of the test instances can be retrieved.

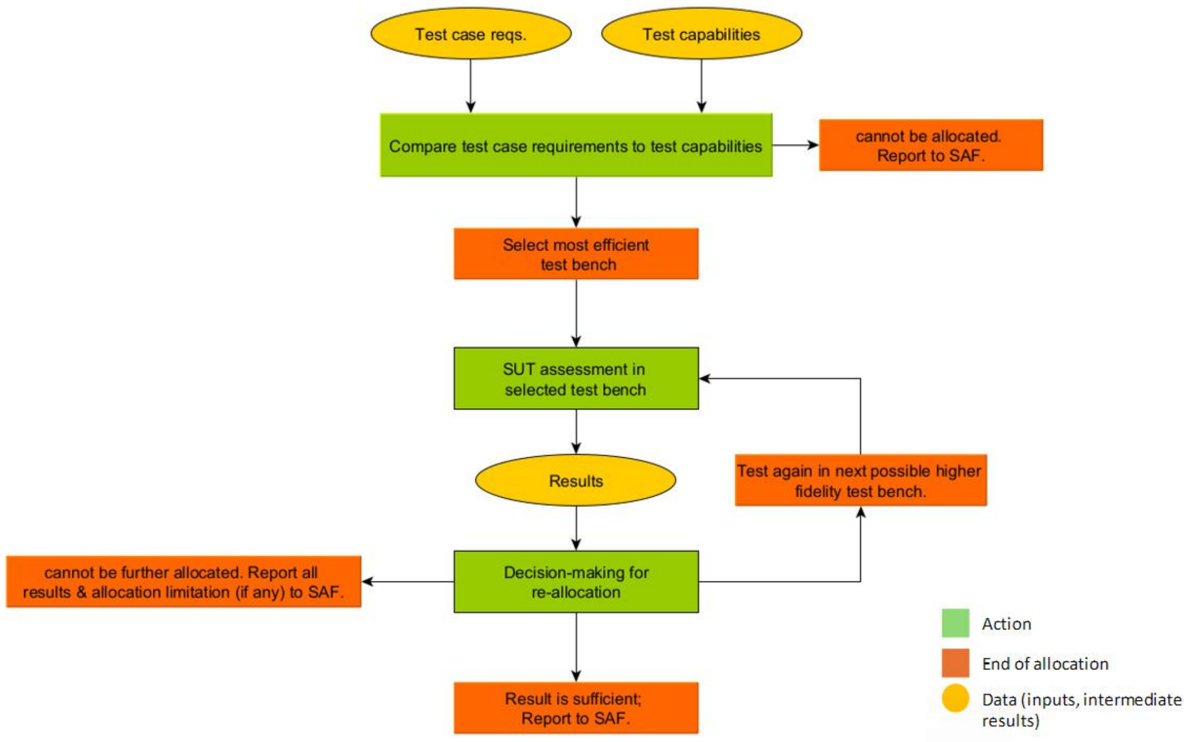

Test case reqs.
Test capabilities
Compare test case requirements to test capabilities
cannot be allocated. Report to SAF.
Select most efficient test bench
SUT assessment in selected test bench
Results
Test again in next possible higher fidelity test bench.
cannot be further allocated. Report all results & allocation limitation (if any) to SAF.
Decision-making for re-allocation
Result is sufficient; Report to SAF.
Action
End of allocation
Data (inputs, intermediate results)

Figure 4: Overview of the initial allocation process.

The first step in the process is the comparison between the requirements from the test cases and the capabilities of the test instances. The requirements can be extracted from the test case by the structure provided in this document. The capabilities of the test instances need to be provided by the owner of the test instances. The comparison is done by following a defined structure that checks each aspect of the test case. A focus is set on the requirements of the scenario description. They need to be fulfilled to reproduce the scenario with all its details on the specific test instance. This structure is based on the ISO 34503 taxonomy "Road Vehicles - Test scenarios for automated driving systems - Specification for operational design domain" [1]. The main structural elements are:

- Scenery elements
- Environment conditions
- Dynamic elements
- Test criteria

This part of the process is described in more detail in chapter 3.

After that comparison, it is known which of the available test instances is capable of executing this specific test case.

It was decided to use a virtual simulation first approach (see chapter 4.2) for the initial allocation due to the fact that virtual simulations are safety hazard and throughput beneficial. This means that all test cases that can be allocated to virtual testing will be executed using the available virtual simulation test instance. In case there are multiple virtual simulation test instances available, the one with the lowest fidelity, that is capable of executing the test case, will be chosen for throughput reasons.

After the execution, a number of decisions will be made to see if the test results from the simulation are sufficient for the test case, or if it is necessary to execute this test case again in one of the other test instances. This part of the process will be described in more detail in chapter 4. In case of a reallocation, the test instance with the next highest fidelity test instance will be chosen.

Some notes on the initial allocation process:

- It is intended that the organization commissioning the safety validation (e.g. a road authority or consumer testing organization) can overrule the decision-making process and immediately allocate individual test cases to test instances with a higher reliability. This can be done for example in case a road authority wants to validate certain test cases on a proving ground instead of XiL.
- It is also foreseen that the person responsible for the execution of a test case at the proving ground may refuse to execute it due to safety concerns (regarding persons, equipment or infrastructure involved). This must be documented and taken into account in the final assessment.
- A part of the proposed initial allocation process already contains the reallocation of test cases to test instances of higher fidelities. This part overlaps with task 3.5 of the SUNRISE project and should therefore be treated as a first approach. The final reallocation process will be described in deliverable 3.5 “Report on the Dynamic Allocation and Validation of Test Runs” and might differ from this first approach.

3 COMPARISON OF TEST CASE REQUIREMENTS AND TEST INSTANCE CAPABILITIES

3.1 Extracting requirements from concrete scenarios using ASAM OpenLabel 1.0

Goal

According to the definition in SUNRISE a concrete scenario is a parameterised model of the time sequence of scenes (logical scenario) which begins with an initial scene and defined point in time; the behaviour of the main actor (vehicle under test) is not further specified. Concrete scenarios require additional metadata to enable three main actions: (1) structure content into storage or databases, (2) enable searching capabilities, and (3) enable ODD comparisons.

In particular, extending concrete scenarios with metadata related to ODD concepts or statements provides the necessary alignment with test execution stages, where ODD concepts are used as input. Other non-ODD concepts can also be used to tag scenarios, for instance related to the relevance of the scenario to specific test requirements or certification processes.

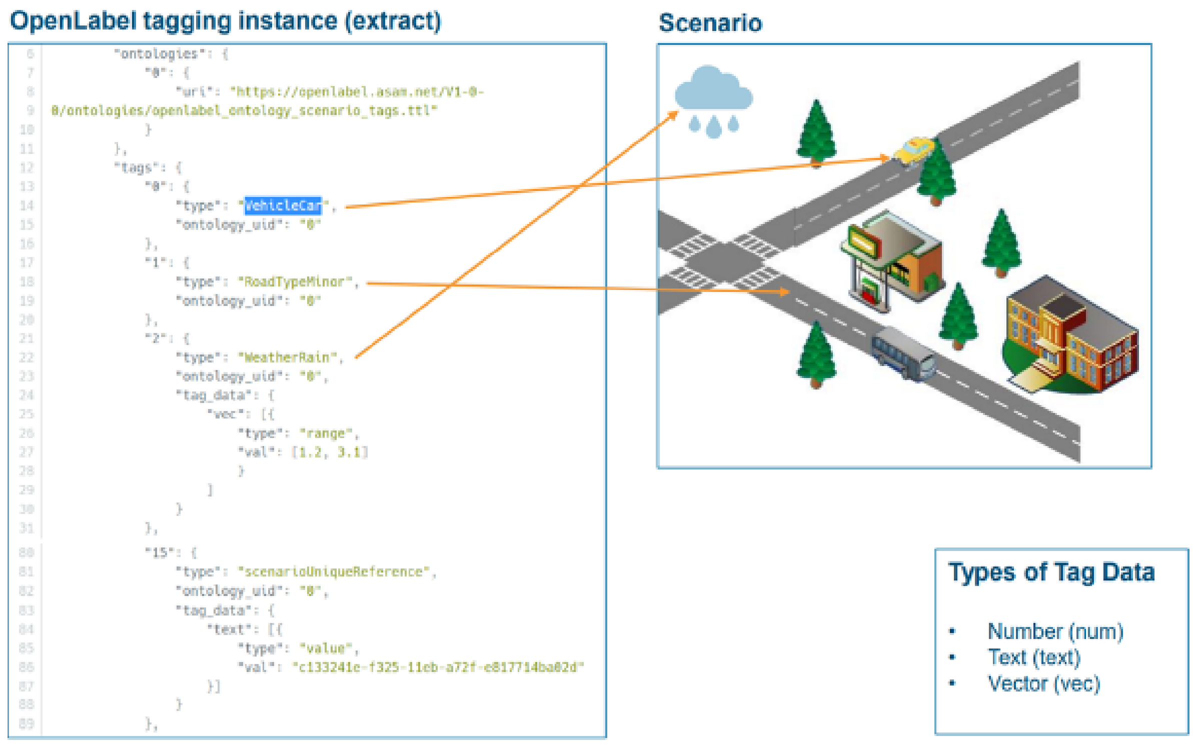
The key goal of the proposed approach in this chapter is to provide a harmonized way to add metadata to the scenario content, so that standards can be used to exploit their existing taxonomies and concepts. In other words, requirements for testing instances can be extracted by reading the standardised metadata files (e.g., formatted as OpenLabel tags) that go with the scenario files (concrete scenarios).

Approach

The proposed approach is to use tags as the atomic unit of metadata to be added to scenario files or data packages. In the ASAM e. V. simulation branch, the OpenLabel 1.0.0 standard provides a comprehensive mechanism to construct such tags and build a file with tags (see Figure 5).

ASAM OpenLabel 1.0.0 has two main parts: one part is devoted to object-level annotations (i.e. specification of annotation to represent sensor-level objects, such as vehicles, pedestrians, etc.), while the other part targets the specification of tags, specially focused on scenario content. Both parts coexist coherently, and the file format is specified with a single JSON schema data model [2].

Tags formatting is flexible and complex enough to support the creation of static and dynamic tags, defined with a name, persistent ID (as well Universal Unique ID), a class type (linked data to an ontology file or URI), and a variety of tag properties, in the form of numerical, categorical or Boolean constructs. These properties can be used to create complex tags that may include ranges, values, thresholds and enumerations.



```
        "ontologies": {
            "0": {
                "uri": "https://openlabel.asam.net/V1-0-
0/ontologies/openlabel_ontology_scenario_tags.ttl"
            }
        },
        "tags": {
            "0": {
                "type": "VehicleCar",
                "ontology_uid": "0"
            },
            "1": {
                "type": "RoadTypeMinor",
                "ontology_uid": "0"
            },
            "2": {
                "type": "WeatherRain",
                "ontology_uid": "0",
                "tag_data": {
                    "vec": [{
                        "type": "range",
                        "val": [1.2, 3.1]
                        }
                    ]
                }
            },
            "15": {
                "type": "scenarioUniqueReference",
                "ontology_uid": "0",
                "tag_data": {
                    "text": [{
                        "type": "value",
                        "val": "c133241e-f325-11eb-a72f-e817714ba02d"
                    }]
                }
            },
```

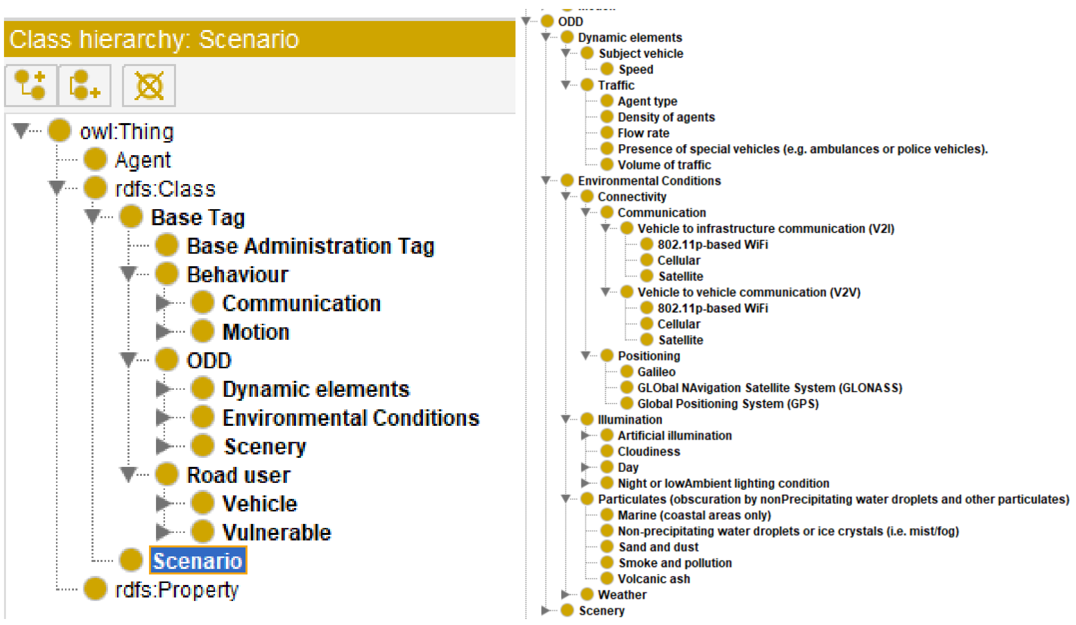
Types of Tag Data

- Number (num)
- Text (text)
- Vector (vec)

Figure 5: Example scenario tagging using ASAM OpenLabel 1.0 tag constructs

The mechanism to produce the data tags is not specified in OpenLabel 1.0, which only focuses on how to format the data tags. This way, the tags can be produced manually or automatically if an algorithm or software can extract or deduce the tag values from the file to be tagged. Semantic rules or logical inference can be used to create scripts which, for instance, read an OpenScenario file, explore the content of the file, discover what objects and actions are defined, and then create a set of tags related to behaviour, ODD, or other scenario-related concepts.

The part of OpenLabel 1.0 related to scenario files is supported by the OpenLabel 1.0 ontology [3]. Figure 6 illustrates part of the OpenLabel ontology, with the expanded tree of tags, inheriting from the ”Base Tag”: children tags relate to terms and concepts (including ODD concepts from BSI PAS 1883 [4]). Nevertheless, the OpenLabel 1.0 specification also allows the user to point to other ontologies freely, so every tag has its own unique class identifier.



Class hierarchy: Scenario

- owl:Thing
  - Agent
  - rdfs:Class
    - Base Tag
      - Base Administration Tag
      - Behaviour
        - Communication
        - Motion
      - ODD
        - Dynamic elements
        - Environmental Conditions
        - Scenery
      - Road user
        - Vehicle
        - Vulnerable
    - Scenario
  - rdfs:Property

- ODD
  - Dynamic elements
    - Subject vehicle
      - Speed
    - Traffic
      - Agent type
      - Density of agents
      - Flow rate
      - Presence of special vehicles (e.g. ambulances or police vehicles).
      - Volume of traffic
  - Environmental Conditions
    - Connectivity
      - Communication
        - Vehicle to infrastructure communication (V2I)
          - 802.11p-based WiFi
          - Cellular
          - Satellite
        - Vehicle to vehicle communication (V2V)
          - 802.11p-based WiFi
          - Cellular
          - Satellite
      - Positioning
        - Galileo
        - GLObal NAvigation Satellite System (GLONASS)
        - Global Positioning System (GPS)
    - Illumination
      - Artificial illumination
      - Cloudiness
      - Day
      - Night or lowAmbient lighting condition
    - Particulates (obscuration by nonPrecipitating water droplets and other particulates)
      - Marine (coastal areas only)
      - Non-precipitating water droplets or ice crystals (i.e. mist/fog)
      - Sand and dust
      - Smoke and pollution
      - Volcanic ash
    - Weather
  - Scenery

Figure 6: ASAM OpenLabel 1.0 ontology structure. ODD terms mostly related to BSI PAS1883 [4]

When the scenario tags are created, the scenario database can be used to store the scenario content with the tags, enabling search queries. These queries can be guided by ODD, test requirements or other concepts, to find, within the scenario database, which scenarios to use for the testing. Figure 7 depicts an example process flow that can exploit a scenario database with scenario tags, using tags formatted as ASAM OpenLabel files, and the ASAM Open Data Services (ODS) (which focuses on the persistent storage and retrieval of testing data) to manage the storage and persistence of test data. In the figure, the flow proceeds as follows: a test description is created, which contains requirements for a scenario, system-under-test, and test method, and stored in the ODS Test Database. Metadata tags can be formatted using OpenLabel standard, so that the query process can then extract scenarios that fit specific test instances' capabilities, SUT requirements, or other requirements. These results can be forwarded to the Test Case preparation process, building the test case. Subsequent execution and analysis stages may feedback results to the input databases if test results are supported.

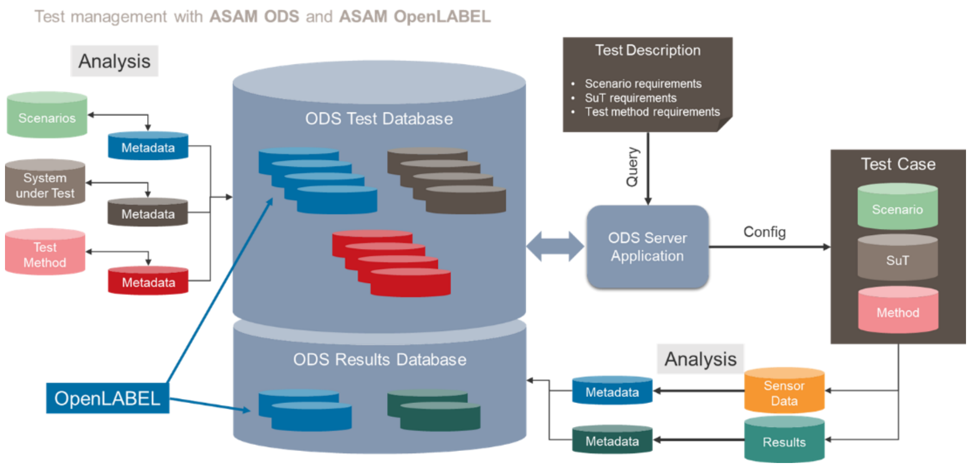


Figure 7: Test management with OpenLabel scenario tags.

3.2 Extracting test instances capabilities

Three types of test instances are part of the scenario-based testing paradigm: virtual simulations, XiL simulations, and proving ground testing. The capabilities of a test instance are to be gathered by a test instance owner. The gathered information must cover all dimensions of the comparison structure which are relevant to a given use case; therefore, aspects not related to the use case may be skipped. For example, UC1.1 in Sunrise is not a connected vehicle use case, therefore information on connectivity need not be gathered for test instances that are planned to be used for UC1.1. **Thus, the first requirement when describing test instance capabilities is to identify the relevant parts of the comparison structure for a given use-case and ensure that the capabilities are described for all relevant parts of the structure.**

The second requirement when describing test instance capabilities is to validate the capabilities. For example, a virtual simulation test instance owner must validate the capabilities of the sensors and vehicle dynamics models that are to be used for testing. For example, to claim that sensor models behave correctly under the environment condition "fog", the sensor models are to be correlated against real-world data with respect to the effect of fog on the resulting (degraded) output of the sensor.

Besides the description of capabilities required for the comparison structure, additional capabilities of the test instance are evaluated in the complete allocation process in chapter 4. **Thus, the third requirement is to quantify the capabilities of the test instance with respect to the metrics considered in the allocation process** (chapter 4.3 for the entire list of metrics). Some examples of these metrics are:

- potential risk to humans or objects who may be part of the test execution.
- the fidelity of the test instance
- the effort required for preparation of the test instance.
- execution efficiency given many tests.

3.3 Structure for the comparison

To make a comparison between the test case requirements and the capabilities of the test instances, a structure is needed that covers all relevant aspects. This structure contains the individual elements required to execute the test case correctly. For each of the elements of this structure, it is necessary to check if the test instance can fulfill the requirements. This means, that all elements in this chapter can be used to define test case requirements and test instance capabilities, and can therefore be used as metrics (see elements in Figure 25) for direct comparison of test case requirements and test instance capabilities by following the defined structure. To make the structure compatible with international standards, the taxonomy of ISO34503 [1] was used where possible. Similarly, as in the standards ISO34503 [1] and BSI PAS 1883 [4] , the structure is organised in a hierarchical way like a tree of elements. This tree contains the elements of ISO34503 [1] and additional elements that the SUNRISE consortium considers as relevant aspects for the initial allocation. Thereby, the elements of the structure serve as a kind of checklist for aspects that should be considered in the comparison of test case requirements to test instance capabilities. In the following figures, the elements from ISO34503 [1] are surrounded by blue boxes as shown in Figure 8. The following subchapters describe the four parts of the structure (see also Figure 8):

1. Scenery elements
2. Environment conditions
3. Dynamic elements
4. Test criteria

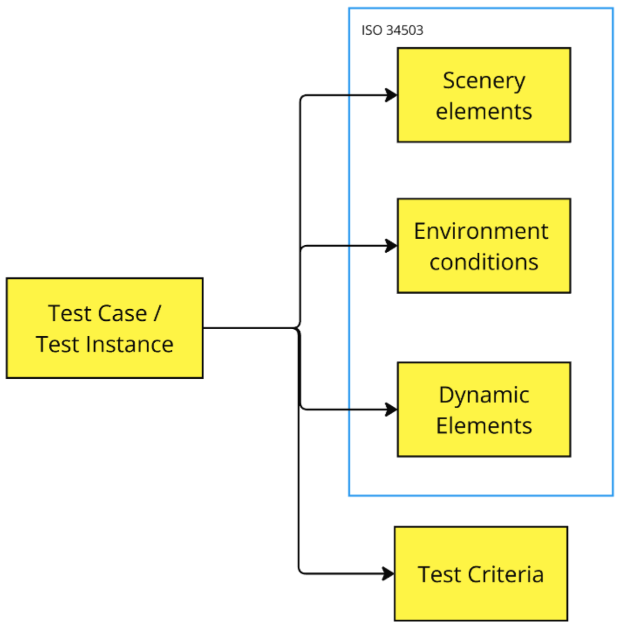


Figure 8: Main nodes of the structure

Important note: To be generally applicable such as ISO34503 [1] and due to the wide variety of test instances and an even wider variety of test cases it is not feasible to provide all details of such a structure for the comparison of test case requirements and test instance capabilities. In Annex 1 are some recommendations provided for the comparison of test case requirements with test instance capabilities using this structure. The structure is shown in Annex 2 for examples of test cases and in Annex 3 for examples of test instances. Annex 4 describes a comparison based on the examples in Annexes 2 and 3.

Please note, that not each element or path of elements is mandatory to describe the test case requirements or test instance capabilities. For example, there might be test cases with no requirements on the environment or virtual test instances not supporting any environment such as esmini or SUMO. In such cases the first element of such a sub-structure of unsupported elements could be marked with "None", “not defined", “not supported” or other appropriate descriptions. It is not recommended to omit them or leave them out, because then it cannot be identified if any of the test case requirements or test instance capabilities are incomplete.

3.3.1 Scenery elements

According to ISO34503 [1] the **scenery elements** are divided into six clusters: the zone, the drivable area, junctions, basic road structures, special structures, and temporary driveable area structures (see also Figure 9). These clusters provide context and help describe the capabilities of the test environment.

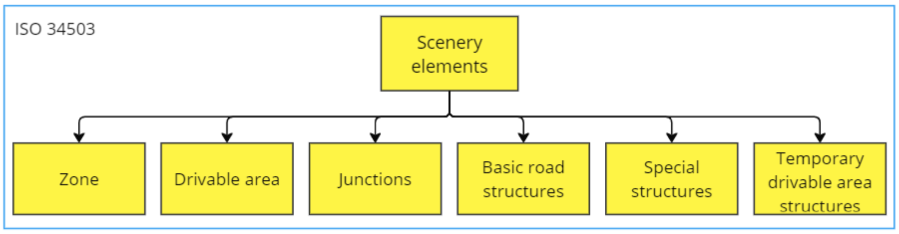


Figure 9: Scenery elements branch of the structure

The **zone** element includes information about special road configurations or areas with specific driving regulations or environmental conditions. Possible attributes are a geo-fenced area, a zone type (e.g. traffic management, school or interference zone), and regions or states, see Figure 10. If a zone is defined, other scenery elements and environment conditions shall be consistent with the zone. For instance, if the zone is a state or a geo-fenced area inside a state where the driving regulations imply left-hand traffic, this should be reflected in the driving directions of the driveable area elements below. Similarly, the environmental conditions in chapter 3.3.2 shall be realistic inside the zone.

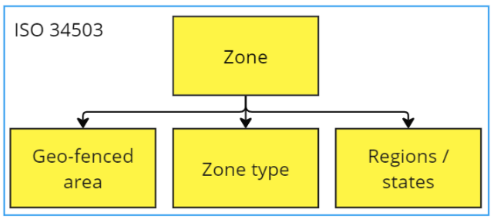


Figure 10: Zone elements branch of the structure

The **drivable area** description is divided into parts for surface (e.g., pavement or friction), signs (e.g., speed limits or traffic lights), type (e.g., motorway, local roads), geometry (like straight or curved), lane specification (e.g., lane markings, direction of travel), edge (like guardrails) (see Figure 11).

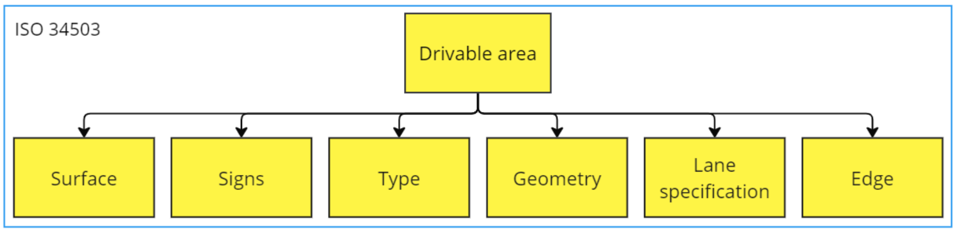


Figure 11: Drivable area elements branch of the structure

The **junctions** element describes areas where two or more roads meet. At least a type (e.g. roundabout, T-junction) and signalization shall be specified and additional attributes can be added if required, see Figure 12.

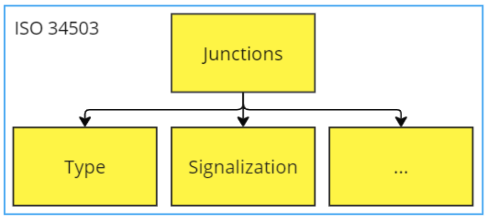


Figure 12: Junctions elements branch of the structure

The description of **basic road structures** includes information on static elements such as streetlights, buildings, and vegetation. Each structure is described by one or more attributes, see Figure 13. Here, at least one attribute should specify the class of the structure (such as streetlight, building, vegetation) or additional attributes should be provided.

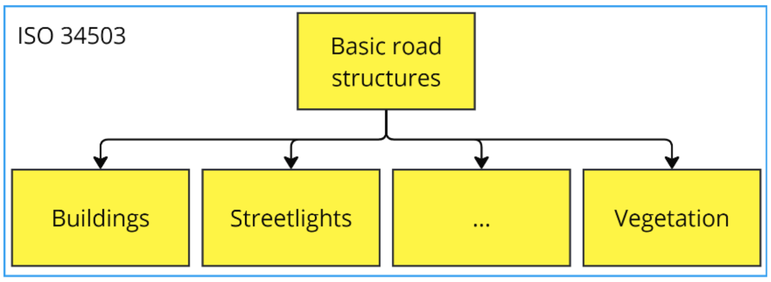


Figure 13: Basic road structures elements branch of the structure

The **special structures** describe special road structures of a specific type like toll plazas, bridges or tunnels with some attributes like dimension (e.g., length, and width). In addition to ISO34503 [1], SUNRISE includes in this category also smart RSUs as type adding further

The special structures element in Figure 9 is detailed in Figure 13. In SUNRISE, this element is extended by an additional branch compared to ISO34503 [1]. In ISO34503, special road structures of a specific type like toll plazas, bridges or tunnels are included in this element with some additional attributes (for details please look into the ISO34503 [1]). SUNRISE included in this category also **smart RSUs** but without the original attributes from ISO34503 (these attributes are coming from the special road structures the smart RSUs are mounted). Instead in smart RSUs blocks elements for sensing and connectivity (Infrastructure-To-Everything) are defined, see Figure 14. Both elements are described in chapter 3.3.3. The sensing element is similar to the one of motor-vehicles, see Figure 19a. The I2X element is similar to the communication interface of the driving function, see Figure 21. Notice that the I2X element must be compatible with its partners, i.e., communication interfaces of vehicles or I2X elements of other smart RSUs.

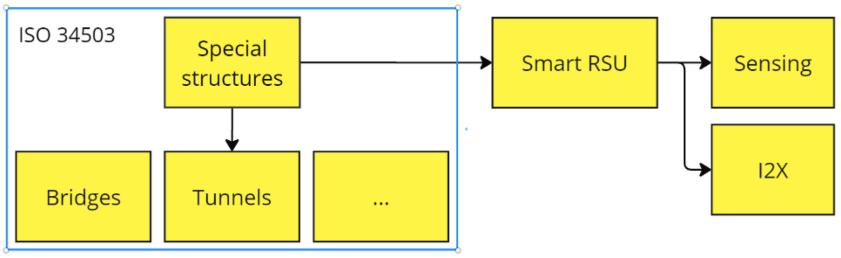


Figure 14: Special structures elements branch of the structure

The last of the scenery elements is the **temporary drivable area** structures, which describe road works, additional signage, or other temporary modifications of the upper elements (e.g., lane offsets), see Figure 15.

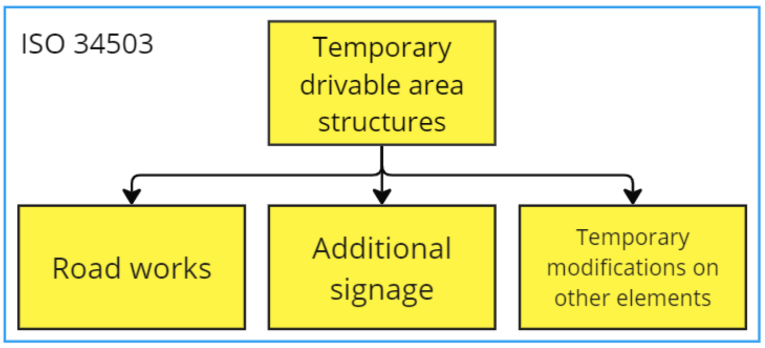

ISO 34503
Temporary drivable area structures
Road works
Additional signage
Temporary modifications on other elements

Figure 15: Temporary drivable area structures elements branch of the structure

To obtain a consistent comparison structure, possible relations between different elements (also from environment conditions and dynamic elements) shall be considered. Some possible relations are indicated rather generally in the standards ISO34503 [1] or BSI PAS 1883 [4]. In the following, some examples are given because there are no concrete examples in the standards. For example, the drivable area is related to the manoeuvrability of dynamic elements because characteristics such as maximum deceleration or lateral acceleration depend on the friction at the surface and slope of the road geometry. Thus, it shall be checked whether prescribed trajectories and manoeuvres of dynamic elements are feasible on the specified drivable area. In case of simulations or virtual environments (with simulated scenery or agents) on the test instance, it shall be checked whether the fidelity is sufficient to represent these effects. Another possible relation between the drivable area and environment conditions, is given in the next chapter. A third example arises when occlusion by a scenery element (e.g., a basic road structure) is part of the test case. Then, the geometry of the scenery element must be represented sufficiently accurate and has to be available to the affected sensor models (a requirement on the fidelity of simulation tools). Additionally, the fidelity of the sensor models shall be sufficient to sense the occlusion.

There can be rare test cases only based on global coordinates (no road or lane references) and without any need of scenery elements for sensing which means that the scenery elements are optional or without any specified requirements on the scenery elements.

3.3.2 Environment conditions

The “Environment conditions” is one of the four blocks of the ODD taxonomy as shown in Figure 8. It is structured according to ISO34503 [1] into four sub-blocks, namely ’weather‘, ’particulates‘, ’illumination’ and ’connectivity‘: the first three blocks represent external factors that can affect the sensor platform of an AV and/or the AV control function during driving (e.g. due to reduced road surface friction coefficient); the last block ‘connectivity‘ refers to the type and characteristics of communications and positioning technology supported by the vehicle as an extra type of external info available to the vehicle. The ISO34503 representation is also considered as a baseline taxonomy in the ASAM OpenODD standard [5], which is under development. The outcome of a test case may be severely affected if adverse weather conditions that affect a specific sensor type are applied, compared to normal conditions, while the support of specific V2X connectivity alternatives may induce a new testing requirement on adding network-relevant test capabilities (e.g. co-simulation with a network simulator). In addition, particulates and illumination are part of environment conditions, equally affecting an ADS perception layer by possibly causing sensor performance degradation for example due to the position of the sun. This subchapter briefly explains each category associated with “Environment conditions (Figure 16).

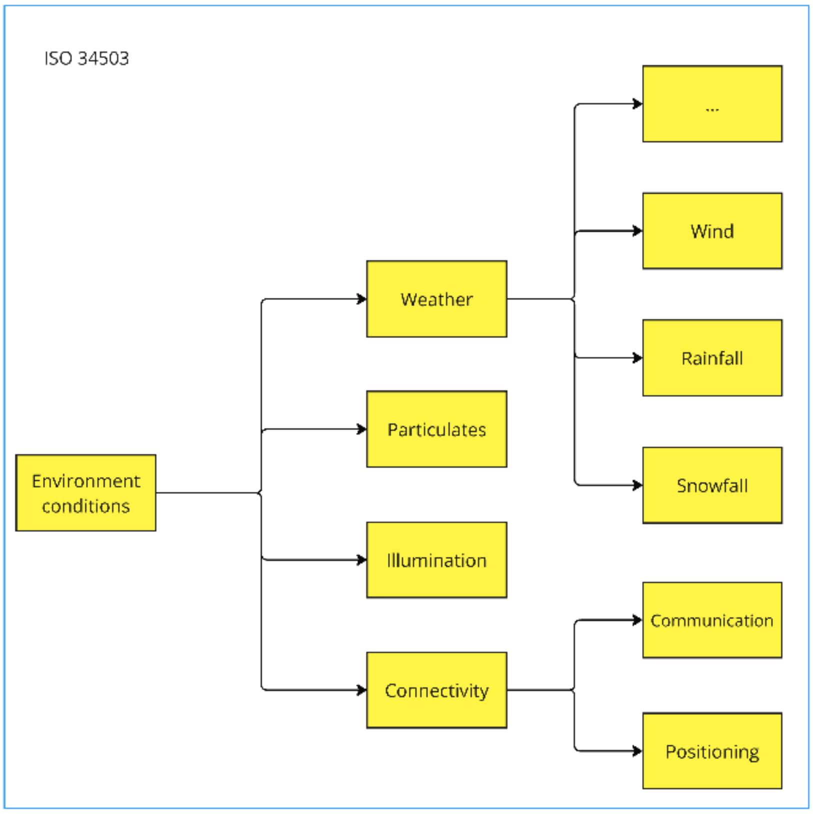


Figure 16: Environment conditions branch of the structure

Weather conditions include ambient temperature, wind, rainfall and snowfall:

- The ambient air temperature may affect sensors and equipment in extreme conditions (low, high), questioning their operational capacity close to the manufacturers’ spectrum. The ODD shall consider such extreme conditions, which may lead to failure-affected test cases. Wind is categorized according to speed and/or direction. Variable thresholds per test case may be applied according to the wind speed, while the direction is classified in general in head, tail and cross wind. For example, vegetation planted adjacent to the route may intervene in the route in strong wind, or even moving shadows may affect reflections upon and ultimately the perception of sensors.

- Rainfall may be measured as the quantity dropped per hour (e.g., mm/h). This phenomenon is quite diverse even in adjacent regions. For a more realistic simulation, average intensity of smaller areas within a minute shall be applied.

- Snowfall intensity is usually arbitrarily classified, mainly based on human perception. Other techniques used for the proper description of the ODD, take into account landscape snow coverage and optics. A combination of concurrent (extreme or not) weather conditions causes extra issues in the simulation within the ODD, thus they should be further analysed through the application of dedicated test instances.

Several **particulates** also affect an ADS behaviour. They include coastal areas, mist, fog, sand, dust, smoke, pollution and volcanic ash. For example, the visibility of the sensors is affected by the accumulation of particulates on the surface of the sensors, reducing their perception capability.

Illumination is another environmental condition that affects the behaviour within an ODD and related test cases. It is classified in day (with attributes like the elevation and the position of the sun), night, cloudiness (during either day or night), and artificial illumination (streetlights, other vehicle lights, other infrastructural lights). The illumination level within a test case is a vital condition, providing totally different capabilities between day and night cases.

Connectivity is divided into two sub-categories, communication and positioning. In both cases, data is transmitted or received between two entities (vehicles, infrastructure, systems or other agents), thus signal strength and interference should be considered while developing a test case. According to ISO34503 [1] also other weather attributes, such as humidity, air pressure, or hail may be part of the test case ODD definition and can be therefore relevant for the definition of test case requirements. Noteworthy, a) for some of the ODD taxonomy elements as proposed by ISO, dependencies to other ODD elements can be observed: for instance, if ODD element rain is set to light rain, then the ODD element road surface should be set to wet. Those dependencies are not stated inside the standard 3.3.3 and shall be taken care of when defining the ODD of a specified test case. b) in practice, it depends on the simulation tool maturity whether one can define all environmental conditions during simulation via a config file. In the case that environment model of the simulation tool does not support the full set of required ODD attributes for the system under test, an alternative would be to create dedicated artificial noise models for specific environmental conditions, applied on the level of emulated sensor outputs so that the effect of a specific environmental condition can be tested without re-creation of such environmental condition during gameplay (as a simplistic example e.g. in CARLA, different noise levels can be applied to GNSS sensor outputs).

3.3.3 Dynamic elements

According to the ISO 34503 [1] standard, the dynamic elements are divided into the **subject vehicle**, which is defined here as motor vehicle, and other **traffic agents** (as shown in Figure 17). These other traffic agents are further classified by their type, including motor vehicles, non-motor vehicles, vulnerable road users (such as pedestrians or bicyclists), animals, and horse riders.

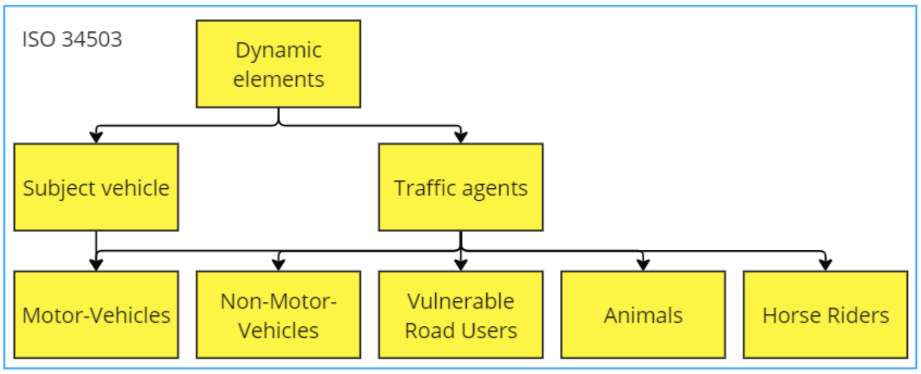


Figure 17: Dynamic elements branch of the structure

Within the SUNRISE project, additional elements have been identified that are not explicitly covered in the ISO 34503 [1] standard but can be used to describe the various traffic agents in more detail. These additional elements further detail **motor-vehicles** and include human representation, driving function, sensing, and vehicle dynamics, see Figure 18. For example, the human representation element could be used to describe the characteristics of e.g., drivers or pedestrians, while the sensing element could be used to describe the use by vulnerable road users of augmented reality or other human-machine interfaces.

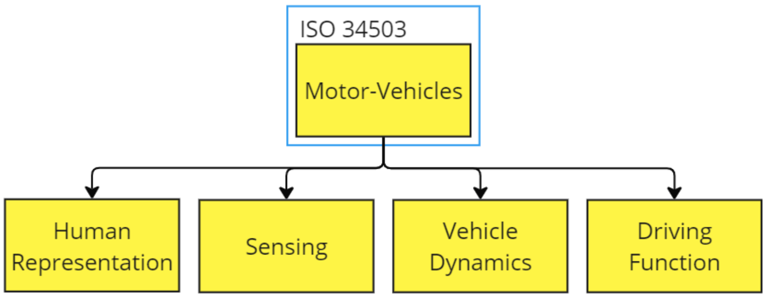


Figure 18: Additional elements branch of the structure outside of ISO 34503 for further detailing motor-vehicles

The **human representation** belongs to the SAE levels 0 to 3 and is divided into a driver or "road user" (see [6]) model for virtual simulations, Simulator (also known as Human-in-the-Loop) used for Driver-in-the-Loop (DiL) or quasi-static Vehicle-in-the-Loop (ViL) simulations (real vehicle on test rig for HMI or sensor tests with sensor penetration), and real human representations (Figure 19) such as on proving grounds including dynamic ViL.

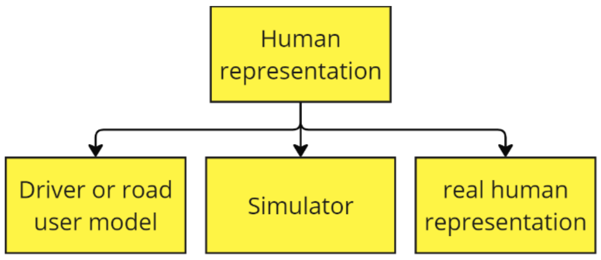


Figure 19: Human representation elements branch of the structure outside of ISO 34503

To ensure the safety of CCAM (Connected, Cooperative, and Automated Mobility) vehicles, it is crucial to accurately simulate human behaviour. This enables the establishment of a baseline against which CCAM vehicles can be tested. Driver models, which are mathematical representations of human behaviour, play a key role in this process. These models can be categorized based on their capabilities and intended applications. Relevant categories for driver models, particularly in the context of safety assurance, have been defined within the V4SAFETY project [7].

The most basic type of driver model is the trajectory follower. This model follows a predefined path without interacting with the environment. It may either strictly adhere to the trajectory or incorporate a controller that attempts to maintain the vehicle on the path while obeying the laws of physics.

Another category is the everyday driving model, which is designed to replicate human behaviour in typical, non-critical driving scenarios. This model is versatile, applicable across a variety of driving situations and environments, and includes features such as car-following, lane-keeping, lane changes, adherence to traffic laws, and navigation, among other functions depending on the model's complexity. An example of an everyday driver model is the ika SimDriver [8].

A more specialized driver model used for safety assurance is the behavioural crash causation model. These models simulate human errors that can lead to crashes, incorporating factors such as inattentiveness, intoxication, drowsiness, and other relevant conditions that could impact driving behaviour. Examples of crash causation are the Stochastic Cognitive Model (SCM) [9] or Driver Reaction Model (DReaM) [10].

Other driver model types for specialized applications do exist. Especially driver models for critical driving scenarios are very relevant for safety assurance. These driver models can also be used as a reference to benchmark the performance of the CCAM vehicle. Examples of driver models used as a reference in critical situations are the Human Driver Performance Model defined in UNECE R157 [11] and the NIEON model developed by Waymo [12].

Similar models can be designed for other road users as well, like pedestrian models ranging from trajectory following models to realistic high fidelity behaviour models for virtual simulations, as shown in Figure 20, with Human-in-the-Loop and real-world testing (e.g., proving ground) being even more realistic options.

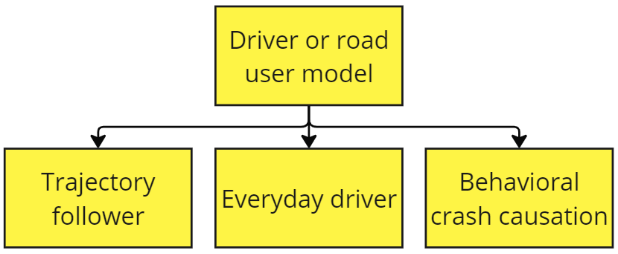


Figure 20: Driver or road user model elements branch of the structure outside of ISO 34503

The **sensing** sub-element (Figure 18) of the dynamic elements is divided into the type of the sensor and its fidelity or sensor effects, see Figure 21. The "Type" block describes the type and characteristics of the sensor (e.g., camera, radar, etc. together with range, direction, and opening angle). The "Fidelity / Sensor Effects" block describes the fidelity and relevant effects (e.g., accuracy, motion blur or laser beam divergence).

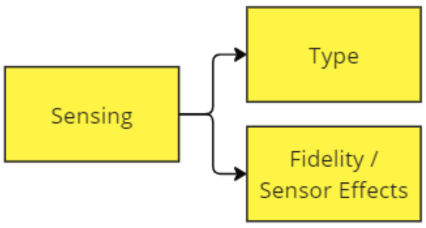


Figure 21: Sensing elements branch of the structure outside of ISO 34503

The **vehicle dynamics** block of Figure 18 can be clustered into tire, suspension, drivetrain, engine, and general, as shown in Figure 22. The general block describes which kind of model may be required, such as single track, double track, multi-body model for physics simulation, or the use of a real vehicle for proving ground testing.

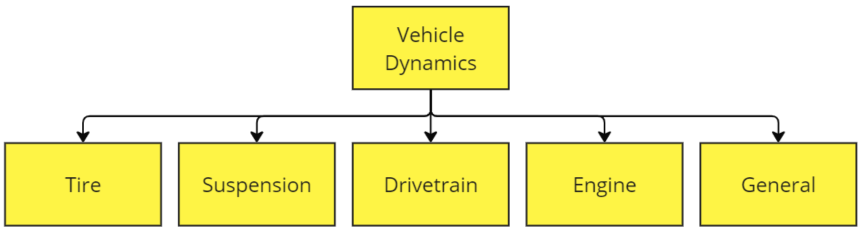


Figure 22: Vehicle Dynamics branch of the structure outside of ISO 34503

The **driving function** itself, in Figure 23, or its capabilities (ODD mainly already covered by upper elements like scenery or environment) are less important for the allocation of test scenarios to the right test instances dependent on their capabilities. However, the interfaces needed for the implementation of the driving function on the test instance to be used, are of high importance. For example, if the driving function needs any kind of raw data from sensors, it could not be tested on a simulation platform only dealing with object list-based sensor models. The Driving Function block is subdivided into interfaces for sensing, communication, positioning, and vehicle dynamics.

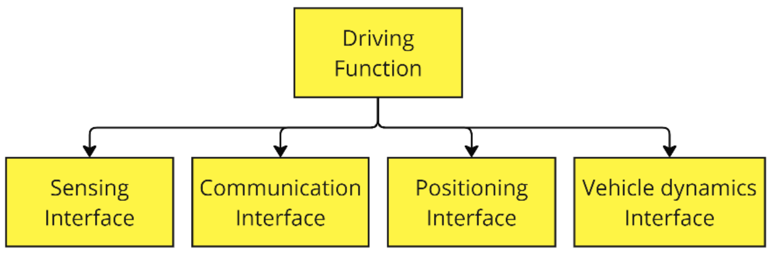


Figure 23: Driving Function with its Interfaces branch of the structure outside of ISO 34503

The **sensing interface** mainly differs in object lists data or raw data dependent on the fidelity of sensor models used.

The **communication interface** describes how a vehicle exchanges information with other agents or smart RSUs. To identify potential incompatibilities between two communication partners, at least the underlying technology and communication content shall be specified.

The **positioning interface** deals with the requirements of the driving function on the positioning information from the test instance, such as GPS/GNSS data or any kind of map information.

The **vehicle dynamics interface** includes all vehicle dynamics data like steering, acceleration, deceleration or braking data (e.g., steering wheel angle, gear, pedal position, clutch, driving speed, tire pressure, torques).

There might be some characteristics (test case requirements and test instance capabilities) related to other ODD elements, such as sensor degradation related to environment conditions (e.g., rain in chapter 3.3.2) or characteristics related to occlusion and some scenery elements (chapter 3.3.1). In case of more detailed sensor models supporting weather conditions or occlusion, the according information must be available to the sensor models (e.g., precipitation from environment for degradation or scenery objects' position and dimensions).

3.3.4 Test criteria

To ensure the safety of CCAM vehicles it is necessary to define test criteria and to show that these are fulfilled. These test criteria need to be defined for the test cases based on the input test requirements. A test process is proposed in [13], further elaborated in [14], and an adopted simplified version focusing on the test criteria element is shown in *Figure 24*. Initial test criteria should be defined as input for the comparison between the test case requirements and the capabilities of the test instances.

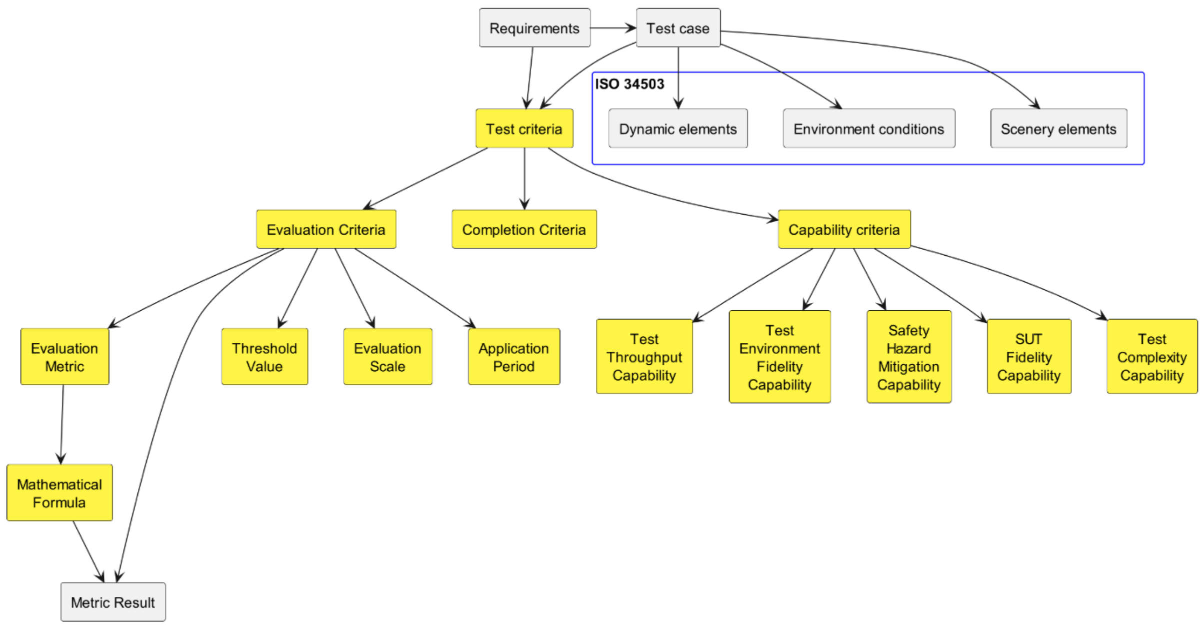


Figure 24. Simplified view of the test evaluation (inspired by [13] [14]).

Test criteria for a specific test case, are usually defined together with clear pass/fail-criteria before the execution of a certain test. Together with pass/fail-criteria, suitable measurable variables are defined which can be recorded during the execution of a certain test and analysed afterwards. These measurable variables shall have sufficient reliability and validity for a pass/fail decision on the specific test case. The input to the test criteria should include safety related requirements based on, e.g., hazard and risk analysis relevant for ISO 26262 and ISO 21448 as well as performance related requirements. Further, all kind of meta information should be included in test criteria that might be useful for the allocation or test execution. These also include meta-information for each test case, such as the test case's purpose (e.g., optimised for sensor testing) or other relevant details (e.g., test case extracted from a crash database indicating collision risks).

The types of test criteria shown in *Figure 24* are:

- **Evaluation criteria.** These are defined such that one or more metric results can be evaluated in relation to defined threshold values or evaluation scales within specified application periods. These metric results are calculated using a mathematical formula (described by an evaluation metric) and data generated during the execution of the test case (test case evaluation data). Evaluation criteria exist at different levels and consequently can be used to evaluate a single scene within a scenario, an entire scenario, a set of scenarios (i.e., a test procedure), or even a set of test procedures [13] [14].

- **Completion criteria:** Conditions under which the testing activities are considered complete [14].

- **Capability criteria:** A number of criteria related to the capability of the testing system:

- **Test Throughput Capability:** The efficiency and speed at which a testing system executes and processes many test cases within a given timeframe. It includes factors such as setup time, execution speed, result analysis, and the ability to handle multiple tests concurrently.
- **Test Environment Fidelity Capability**: The degree to which models accurately replicate real-world environmental conditions, including various scenarios, responses, and behaviours of the vehicle and other road users applicable for the test coverage item.
- **Safety Hazard Mitigation Capability**: The potential danger and uncertainty associated with track testing involving physical vehicles and obstacles, posing risks to test participants such as safety drivers and experiment observers applicable for the test coverage item.
- **SUT Fidelity Capability**: A metric designed to quantify the effects of abstraction between a model and its intended production implementation, considering the potential limitations of virtual environments or test harnesses applicable for the test coverage item.
- **Test Complexity Capability:** The level of intricacy involved in conducting track testing, considering the ability of testing facilities to accommodate diverse test elements and ODD conditions applicable for the test coverage item.

After the allocation of test cases to test instances, the test criteria may need to be refined to adhere to the used test instance.

4 METRICS AND DECISION MAKING

Typically, trade-offs exist that must be considered when allocating test cases to the main test instance types: virtual simulation, XiL, and proving ground. For example, test instance fidelity and test throughput are often inversely proportional, due to the additional efforts required to execute test cases in high-fidelity test instances. In proving ground testing which relies on real vehicles, there are additional constraints of increased risk to personnel and equipment during testing as well as reduced complexity of scenarios that can be tested.

Given the large number of test cases required for a strong safety argumentation, it is important for the decision-making process to prioritize test throughput and reduce test effort, where possible. It is equally important to ensure that the test results are reliable and provide confidence about the safety of the system under test. Both these requirements must be addressed by the allocation process. To address both requirements, the initial allocation process may require decision-making based on test results obtained by executing the test cases. Although this appears as a "re-allocation" step and may overlap partially with D3.5, it is a necessary step to obtain key knowledge about the safety relevance of the test cases and to understand the limitations of the test instances. This knowledge is necessary to judge when test cases must be allocated to higher fidelity test instances, which although have high test effort and costs, are necessary for reliable safety assessment of the SUT.

Chapter 4.1 provides an overview of the state-of-art with respect to the allocation of test cases to different test instances, especially focusing on how the allocation process may deal with the fidelity limitations of virtual simulation and the subsequent uncertainty of test results.

Chapter 4.2 describes the virtual simulations first approach and how it relates to some of the metrics described in chapter 4.4.

Chapter 4.3 describes different metrics to be considered when comparing test case requirements with test instance capabilities using the structure proposed in Chapter 3.

Chapter 4.4 discusses a broad range of metrics that can be considered for a general initial allocation methodology of test cases to test instances. 4.3

Chapter 4.5 expands on the decision-making process shown in Figure 4 (chapter 2)**,** detailing the key decision points in the process. The decision-making incorporates the learnings from the state-of-the-art and utilizes metrics presented in chapters 4.3 and 4.4 to compare the test case requirements with test instance capabilities. It also incorporates iterative allocation and testing, given the fidelity limitations of virtual simulations, to ensure the test results are reliable.

Finally, Chapter 4.6 outlines the main outputs of the initial allocation process in order to facilitate not only the testing of the SUT, but also to enable more elaborate decision-making on reallocation and further testing (see D3.5 of SUNRISE) and safety argumentation as needed for the SAF (see D2.2 of SUNRISE).

4.1 State-of-the-art of test case allocation to test instances

This chapter reviews the state of the art with respect to allocation of test cases to different test instances, especially focusing on how the allocation process may deal with the fidelity limitations of virtual simulation test instances and the subsequent uncertainty of test results. As mentioned in the introduction of chapter 4, this is essential for large-scaling reliable assessment of the SUT.

Two main themes stand out in the literature review which are:

1. Prioritize further test execution based on simulation results
    a. Identify safe and unsafe regions of scenario parameter space and sample from each region, in addition to evaluating marginally safe cases. [15]
    b. Identify performance boundaries of SUT, and sample scenarios from performance boundaries. [16]
    c. Perform Transfer Importance Sampling to estimate favourable parameter distribution for sampling scenarios and estimating failure likelihood. [17]
    d. Investigate how well different performance metrics transfer between simulation and real-world when testing a deep neural network-based AV controller, e.g., metrics evaluating *controller behaviour, those quantifying uncertainty in the controller*. [18]
2. Prioritize further test execution based on limitations of the simulation tool.
    a. Assign test cases to test instances based on validity of test instance configuration and testing costs [19]
    b. Using knowledge of the gap between simulation and real-world, or any known limitations of the simulation platform. [15] [18]

The literature review makes it evident that we must consider both the simulation results as well as (known) limitations of the simulation tool in our allocation process to recognize when test results are unreliable and are to be repeated in higher-fidelity test instances. In particular, test cases at performance boundaries – tests in which the SUT barely meets the safety criteria, can be important to test further. [18] illustrates that various safety metrics may be considered here. Furthermore, regions of low validity of the lower fidelity test instance must be identified and tests within these regions shall be repeated.

A topic that is considered important for the allocation process in the deliverable but was not found to be addressed by the current state-of-art is the consideration of safety standards in the allocation process. Safety of the intended functionality (SOTIF) is currently one of the most important safety standards around scenario-based assessment. Consideration of potential triggering conditions or functional insufficiencies of the SUT, as per SOTIF standard, are not considered for the allocation process. Note that the ISO 26262 functional safety standard is aimed at SW/HW faults and therefore does not contain many guidelines for scenario-based testing.

4.2 Virtual simulation first approach

The **virtual simulation first approach** was first introduced in chapter 2 and prioritises test instances with higher throughput metric (chapter 4.3) which is equivalent to higher number of virtual components compared to real components. Especially, virtual simulation test instances have higher priority than proving ground tests. The approach aims at maximizing throughput for test cases when possible. Test throughput is closely related to low test effort and costs and enables large-scale testing.

If two test instances fulfill all test case requirements and have the same or very similar level of virtual components, the one with higher throughput (chapter 4.3) gets higher priority.

This approach is covered or related to many more metrics from chapters 4.3 and 4.4. The virtual simulation first approach in general is similar to the low test complexity metric (chapter 4.4) since usually a lot of models are used that represent a simplification of the real world. In general, the virtual simulation first approach is also similar to low implementation effort as described in chapter 4.4. However, requirements for high-fidelity models can increase the implementation effort and test execution time or decrease the test throughput a lot. Since the main application area of the SAF is the safety assessment, the highest priority is given to the safety hazard metric (chapter 4.3) which is also covered by the virtual simulation first approach due to the lower safety hazard with an increasing number of virtual components.

This approach does not automatically mean that the test instance with the highest priority must be a virtual simulation test instance. According to chapters 4.4 and 4.5 all test instances must first fulfill the test case requirements which could include a need for hardware components, such as high-fidelity requirements or hardware performance test as part of the test criteria.

This approach is based on throughput, fidelity, safety hazard and costs metrics and giving higher weights to them according to the metric application from chapter 4.4.

4.3 Metrics for comparison of test case requirements to test instance capabilities

This chapter describes metrics for comparing test case requirements to test instance capabilities that can be derived from structure in chapter 3.3. The goal of metrics is to enable a sorting of available test instances according to how well each one fulfills the requirements of the current test case. Due to the wide variety of test cases and the different aspects that must be considered in the initial allocation, it is not possible to provide a precise formula for a metric that fits every test case. This chapter describes typical metrics for functional aspects of the initial allocation process that can be derived from the structure in chapter 3.3. Depending on the current test case, these individual metrics can then be combined (also with metrics from the next chapter) into a tailored metric that measures how well a test instance meets the current requirements. A suggestion on how a tailored metric could look like is given at the end of this chapter.

The main metric for the comparison of test case requirements with test instance capabilities is **fidelity**. The fidelity metric also covers a kind of availability metric. This means in case of no defined requirements for an ODD element the fidelity can be set to “None", "N/A" or other explaining values. In case of unsupported ODD elements from the test instance, the fidelity metric can also have values such as “None", “N/A", and others. The fidelity can be expressed by a degree of realism or by the type of models/real components but can also be described in more detail by physical quantities e.g., required or available rainfall intensity or covered accelerations. Additional information to derive the fidelity of single elements can come from the test criteria (see chapter 3.3.4), such as the purpose of the test case (e.g., perception or performance tests) influencing the fidelity of single scenery elements, environment elements or sensor models or the need for real components instead of models.

All **physical quantities** can be used as metrics for the comparison of test instance capabilities and test case requirements, typically dimensions, degree of freedom, movements of dynamic elements (e.g., dimensions, positions, velocities, accelerations; translational and rotational), rainfall intensity, brightness or ambient air temperature for environment and dimensions and positions of scenery elements. These metrics can be derived from scenario description but can also be strongly related to the fidelity metric or in other words, the fidelity can often be expressed by such physical quantities in more detail as stated above.

One other metric for the comparison of the elements is the **test complexity** metric for the test case requirements or the test instance capabilities. E.g., the number of dynamic elements from the test case requirements can be compared to the maximal number of simultaneously moving agents or agents that can simultaneously be detected by sensor models.

Single elements can be used to apply the **safety hazard** metric for the comparison of test case requirements and test instance capabilities. Some of these elements might be the absence of junctions or crossing trajectories indicating any potential collisions or real human representation indicating involvement of human beings, as well as the general sub-element of the vehicle dynamics element indicating the involvement of physical vehicles. In the case of test case requirements, the source of the test case (e.g., police reported crash or in-depth crash database) or any result of a risk assessment as a potential part of the test criteria can be a basis for the safety hazard metric.

The safety hazard metric can also be used to describe the test instance capabilities. In general, the higher the number of virtual components the lower is the safety hazard of the test instance. However, the safety hazard of HiL and SiL test benches is very similar and the safety hazard of ViL and proving ground testing strongly depends on the test case and its requirements (e.g., need for real vehicles, real humans as pedestrians or drivers).

The element of real human representation describing the test case requirements might result in the need for real-time application. This could be directly compared to a **real-time capability** of a test instance. Another option would be to compare the simulation time from the test criteria (see chapter 3.3.4) with the test execution time based on test throughput and the simulation time.

As mentioned in the introduction of this chapter, the **goal** is to sort test instances according to how well they meet the requirements of a given test case. One possibility to measure how well a test instance fits to a test case is combining the individual metrics presented above and in the next chapter into a **tailored metric** that reflects the importance of each individual aspect. In the following, a suggestion for such a tailored metric is given. One of the simplest forms of such a combination is

$$m(i) = sign\left(\prod_{j\in J} m_j(i)\right)\cdot\left(\sum_{j\in J} w_j \cdot m_j(i)\right) \quad \Bigg| \quad \begin{array}{c} \sum_{j\in J} w_j = 1 \\ 0 < w_j \leq 1 \;\forall\, j \in J \\ 0 \leq m_j \leq 1 \;\forall\, j \in J \end{array} \qquad (1)$$

or equivalent

$$m(i) = \begin{cases} \sum_{j\in J} w_j \cdot m_j(i)\, 0 & 0 < m_j \;\forall\, j \in J \\ 0 & else \end{cases} \quad \Bigg| \quad \begin{array}{c} \sum_{j\in J} w_j = 1 \\ 0 < w_j \leq 1 \;\forall\, j \in J \\ 0 \leq m_j \leq 1 \;\forall\, j \in J \end{array} \qquad (2)$$

where i is the index or label of a test instance, j is the index or label of an individual aspect (e.g., one of those mentioned above), m_j is the metric value corresponding to aspect j, w_j is a weighting factor representing the importance of aspect j, and J is the set of indices or labels. The metrics $m_j(i)$ are binary or continuous functions with values larger than 0 up to 1 indicating that test instance i fulfills requirement j and 0 that it does not. Then, the product acts as a kind of fulfillment factor, i.e. it is 0 if a requirement is not fulfilled and 1 if all requirements are fulfilled. The condition $m_j(i) \geq m_j(k)$ means that test instance i performs better than test instance k with respect to the aspect j. If the opposite shall be achieved that $m_j(i) \geq m_j(k)$ holds for the metric m_j but test instance i performs worse than test instance k with respect to the aspect j then introduce a new metric $m_j^* := 1 - m_j$. Then, the sum measures how "desirable" a test instance is for the test case because it accumulates the metrics for $j \in J$ weighted by their importance. Consequently, $m_j(i) = 0$ means that the testcase is not allocable on test instance i, and $m_j(i) > m_j(k)$ means that test instance i is better suited for test case than test instance k. This enables the afore mentioned sorting of test instances (decreasing order of total metric scores from highest to lowest values).

4.4 Metrics for decision making

A decision-making process can include many metrics for the initial allocation. All metrics from chapter 4.3 should be part of a decision-making process since the identification of capable test instances should always be the first step of decision-making of an initial allocation process which means that the test case requirements have to be considered.

The metrics of chapter 4.2 compare all test case requirements to the test instance capabilities. However, a decision-making process can include more metrics concentrating on the test instance capabilities and therefore on the comparison of test instance capabilities between each other (see Figure 25).

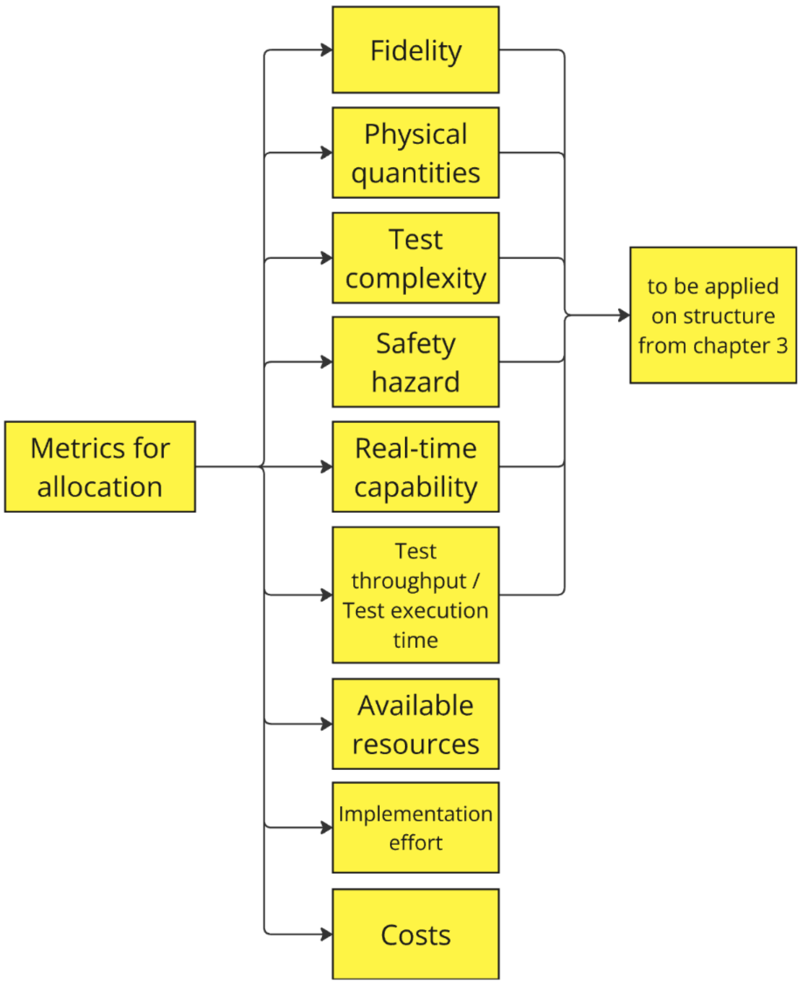


Figure 25: Metrics that can be used for the (initial) allocation of test cases to test instances

One of these metrics is the **available resources** metric which means the general availability of resources, but also the current availability of experts for test execution and equipment not in maintenance. The general availability of resources is strongly related to the costs metric (see below) and test throughput metric (see chapter 3.3.4). Test instances of lower costs such as virtual simulation test instances are usually available in large numbers and have therefore a high test throughput (virtual simulations are usually much faster than real-time and many simulations can be executed in parallel).

A very important metric is the **implementation effort** of the test case which is sub-divided into three sub-metrics which are:

- the implementation or test instance integration effort of the System under Test (SUT),

- the implementation effort of the test case itself and,
- a potential parameterisation or validation effort for the usage of models.

While for proving ground testing (including ViL) and partly for hybrid test instances (HiL or DiL) the integration effort of the SUT can be quite low (due to already existing hardware e.g., ECUs or prototype vehicles), it is very likely that the implementation effort of the test case is very high (e.g. right positioning of vehicles or robots). The parameterisation or validation effort is low (e.g. programming of robots) due to the usage of real components instead of models (increases with the number of virtual components e.g., ViL or HiL). But the implementation effort in total strongly depends on current test case setup and the setup of the test instance from prior test cases. In total one can expect an increasing implementation effort with the increasing realism or increasing number of real components.

Due to many existing standards such as ASAM OpenSCENARIO, OSI or FMI the implementation effort of the test case into virtual simulation test instances is usually low. However, the effort for the integration of the SUT or other models can be high due to the used or supported interfaces, the number of signals/interfaces to be connected and the number of signals to be logged. In general, the total implementation effort gets low with an increasing number of virtual components or a decreasing number of real components. However, a large number of high-fidelity models (e.g. graphics card based physical models) causes high efforts for parameterisation and validation of these models and therefore may be higher implementation effort compared to tests of real components as in hybrid or real-world testing.

The **costs** metric shall consider the total costs of the test case to be executed on the test instance e.g., including the costs for implementation. It is strongly related to other metrics, such as throughput, available resources or fidelity. Usually, test instances of low costs are available in large numbers and have a low fidelity and therefore high throughput.

Additional metrics can be obtained from the test criteria described in chapter 3.3.4 and especially the five-denoted *Capability criteria* for the comparison of test instances. In a real test system, it is common for high performance in one of these criteria to be accompanied by lower performance in other criteria. A fictive example of the dependency is shown in *Figure 26*. This figure aims to illustrate how prioritizing a high level for one of the capabilities, may have negative effect on another one, and consequently becomes a test design choice that also may have an impact on the test instance selection:

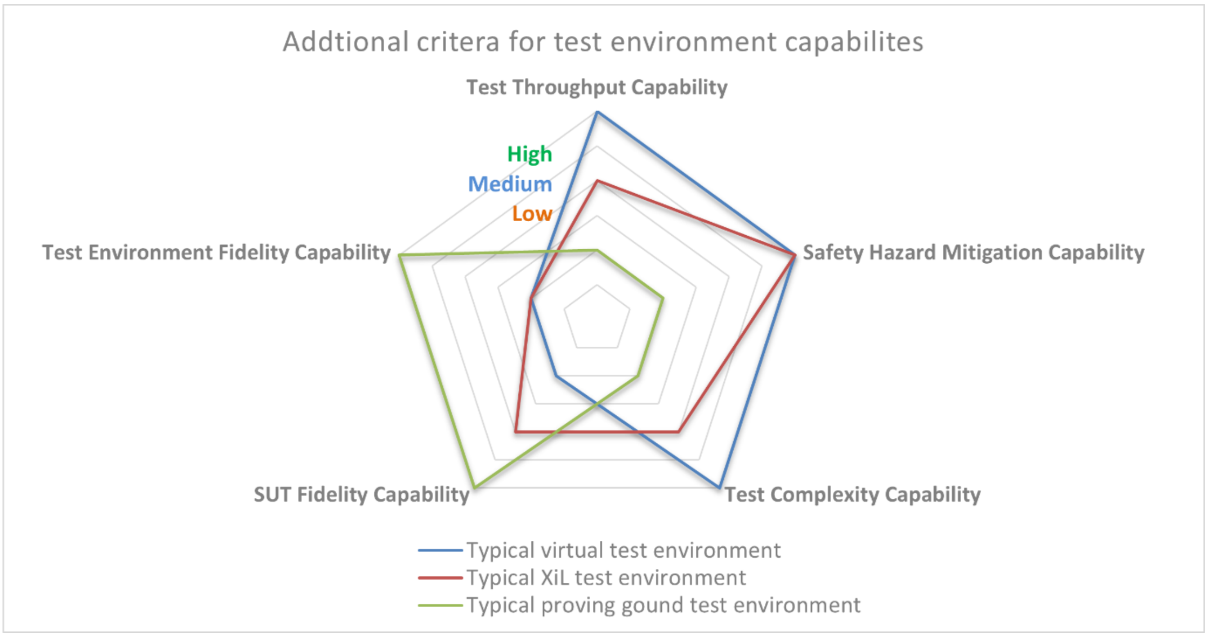


Figure 26: Fictive example test environment capabilities.

When evaluating the capabilities of a typical virtual test environment, it is frequently noted for its high-test throughput capability due to its ability to execute faster than real-time and run instances in parallel. Its safety hazard mitigation capability is generally high, as no kinetic energy is involved, though ethical considerations must be addressed in DiL tests. The test complexity capability is substantial, constrained primarily by the extent of scenario modelling efforts. However, virtual test environments typically demonstrate low SUT fidelity capability, often employing highly abstracted models lacking the emergent behaviour of integrated target SW and HW. Additionally, they tend to exhibit low test environment fidelity capability, indicating that the precision and realism of the testing conditions may be limited. An example of a typical virtual test environment evaluation can be seen in blue in *Figure 26*.

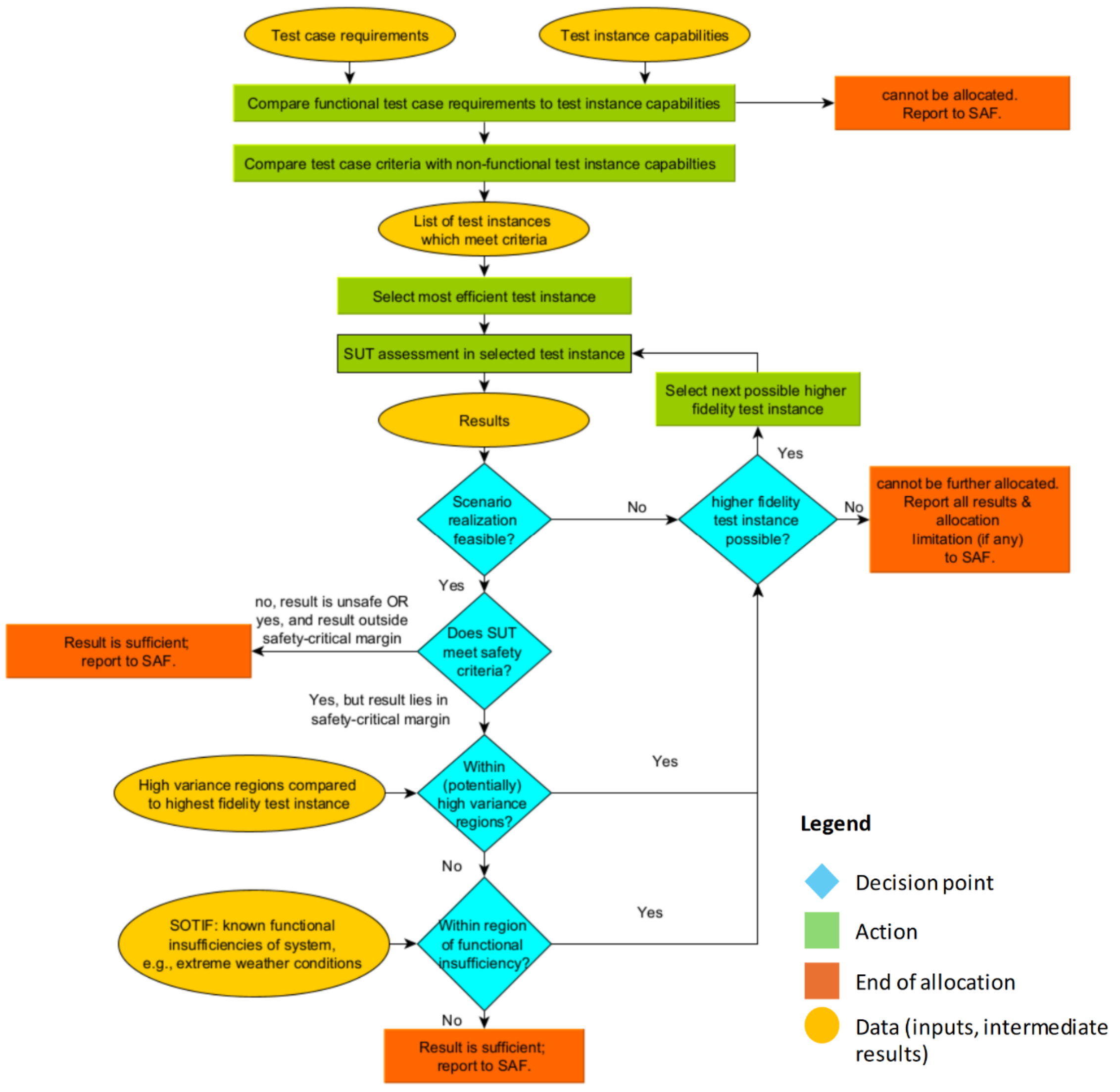
When evaluating the capabilities of a typical XiL test environment, it is observed to possess medium test throughput capability as it runs on target HW. The safety hazard mitigation capability is high, as the virtual components of the XiL framework significantly reduce the risk associated with physical testing. The test complexity capability is often medium, suggesting that while the environment supports a range of testing scenarios, it may be limited by integrating real and virtual components. The SUT fidelity capability is similarly rated as medium, indicating a reasonable but imperfect replication of production target SW and HW within the test environment. However, the test environment fidelity capability is low, as most test harnesses in XiL environments are geared towards testing just one specific aspect of the system, often linked to HW-SW integration testing, which points to limitations in accurately replicating the real-world conditions and dynamics within the XiL setup. An evaluation example of a typical XiL test environment evaluation can be seen in red in *Figure 26*.

When evaluating the capabilities of a typical proving ground test environment, it is noted to have low test throughput capability. This limitation arises from the time-intensive nature of setting up and conducting each test scenario physically, which inherently restricts the volume of tests that can be conducted simultaneously or sequentially. The safety hazard mitigation capability is also low, as the physical testing involves real vehicles and environments, thereby increasing the risk of damage or injury compared to virtual or semi-virtual environments. Similarly, the test complexity capability is low, restricted by practical limitations in orchestrating complex scenarios that might be more easily implemented in virtual settings.

Conversely, the proving ground test environment excels in SUT fidelity capability, which is high due to using real systems in real-world conditions, ensuring an accurate representation of vehicle behaviour and performance. The test environment fidelity capability is similarly high, benefiting from the real-world setting that faithfully replicates the operational conditions under which the vehicle systems are intended to function. However, care must be taken when using dummy test objects as they might miss real-world fidelity crucial to the test objective. The general high fidelity comes at the cost of flexibility and safety in testing. An evaluation example of a typical proving ground test environment evaluation can be seen in green *Figure 26*.

4.5 Decision making for test case allocation to test instances

This chapter details the proposed process for the initial allocation of test scenarios to test instances. The proposed decision-making process is shown in Figure 27. The decision-making process includes the comparison of test case requirements and test instance capabilities to evaluate suitable test instances for test cases. Furthermore, it proposes a method to address the trade-off between efficiency of test execution and reliability of the test results, by testing selected scenarios in higher fidelity test instances. This enables comprehensive (large number of test scenarios) as well as reliable assessment of the SUT. As mentioned in the introduction, achieving this assessment of the SUT requires iterative test instance selection in the initial allocation process, as the safety-critical relevance of the test case and the uncertainty of test results in a given test instance, can only be assessed based on test case results. Note that the proposed approach in this deliverable is the **initial allocation process** and may be extended by D3.5, where the final allocation process will be outlined. D3.5 may propose additional re-allocation strategies for test scenarios to tackle safety-critical aspects such as incorrect execution of tests, or incorrect identification of test capabilities.



Test case requirements
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 capabilties
List of test instances which meet criteria
Select most efficient test instance
SUT assessment in selected test instance
Select next possible higher fidelity test instance
Results
Yes
Scenario realization feasible?
No
higher fidelity test instance possible?
No
cannot be further allocated. Report all results & allocation limitation (if any) to SAF.
Yes
no, result is unsafe OR yes, and result outside safety-critical margin
Result is sufficient; report to SAF.
Does SUT meet safety criteria?
Yes, but result lies in safety-critical margin
High variance regions compared to highest fidelity test instance
Within (potentially) high variance regions?
Yes
No
SOTIF: known functional insufficiencies of system, e.g., extreme weather conditions
Within region of functional insufficiency?
Yes
No
Result is sufficient; report to SAF.
Legend
Decision point
Action
End of allocation
Data (inputs, intermediate results)

Figure 27: Decision-making process for initial allocation.

As the first step, a comparison of the test case requirements with the test instance capabilities is performed, using the approach outlined in chapter 3 and the metrics in chapters 4.3 and 4.4. For example, if the purpose of the test case is to test the perception module of an AV stack, this suggests the need for a high-fidelity environment and sensor simulation. Other metrics such as the complexity of test execution or safety hazard posed during test execution are also considered during this step.

Once this step is complete, a list of suitable test instances for the test case is identified. The remainder of the allocation process is guided by **two principles**: the **"virtual simulation first approach"**, and the **identification of tests that should be repeated in higher fidelity test instances** to ensure reliability of test results. Together, these principles help address the trade-off between efficiency of test execution and the reliability of test results and thereby enable large-scale reliable testing of the SUT.

Following the **"virtual simulations first approach"** from chapter 4.2 virtual simulations (which are usually the most efficient test instance) are prioritized for the allocation when possible. Next, we must **identify which tests are important to repeat in higher fidelity test instances**. The following decision points are proposed in the allocation strategy when deciding whether to repeat a test in a higher fidelity test instance (see Figure 27):

1. Feasibility and correctness of test execution: During test preparation or test execution, it may be found that the test cannot be accurately performed on the test instance. In addition, validation checks are to be made to ensure that test execution is performed correctly. If the scenario execution is deemed inaccurate or infeasible, then further testing is necessary.

2. SUT behaviour evaluated against safety criteria: If SUT behaviour fails to meet the safety criteria in a test or meets the safety criteria by a large margin, the results may be deemed sufficient for the safety assessment. However, given the uncertainty of lower-fidelity test instances, it is risky to conclude this for a scenario where the SUT behaviour is safe but still lies within a safety-critical margin. The complexity of the real-world may not be well-represented; thus, the SUT behaviour can deviate in real-world and become safety-critical. This is especially important when other safety-relevant factors are involved, as below.

3. The scenario lies in a (potentially) low-validity region of the test instance: Such low-validity regions can be identified based on correlation studies across test instances, validation reports for the test instance, or generally accepted limitations of a test instance. For instance, it is commonly accepted that the perception outputs of a SUT can differ between simulation and real-world conditions. Thus, a scenario in a low-validity region may be further tested, especially when SUT behaviour is found to be within a safety-critical margin. When limited information is available on where the low-validity regions lie, then a large low-validity region may be assumed.

4. The scenario contains a potentially triggering condition that exposes functional insufficiencies of the system, as per the safety standard SOTIF: A SOTIF analysis of the SUT can reveal potential triggering conditions and functional insufficiencies of the system. As these are safety-critical regions, these shall be well-tested across test instances, to ensure that the SUT behaves safely in these regions, especially when the SUT behaviour is found to be within a safety-critical margin.

Once a test is identified to be repeated in a higher-fidelity test instance, the allocation strategy allocates the test to the most effective higher-fidelity test instance. For example, suppose the available test instances that meet the technical comparison include a low-fidelity simulation toolchain, a high-fidelity simulation, XiL test instance, and a proving ground. A test is most effective to perform on a low-fidelity simulation toolchain, followed by the high-fidelity simulation, XiL, and finally proving ground. Therefore, the test is first assigned to the low-fidelity simulation toolchain. Now, if deemed necessary to be repeated, it is allocated to the next effective higher fidelity test instance; the high-fidelity simulation toolchain. Once the test is performed, the same decision points as above can be used to determine if the test execution is now sufficient for the safety assessment or whether further testing is needed. This iterative process is continued till the results are deemed sufficient for the safety assessment, or test is executed in the highest-fidelity test instance possible and cannot be further allocated. Any limitations with respect to the allocation shall be reported to the SAF. The comprehensive reporting of allocation results to the SAF is discussed in chapter 4.6.

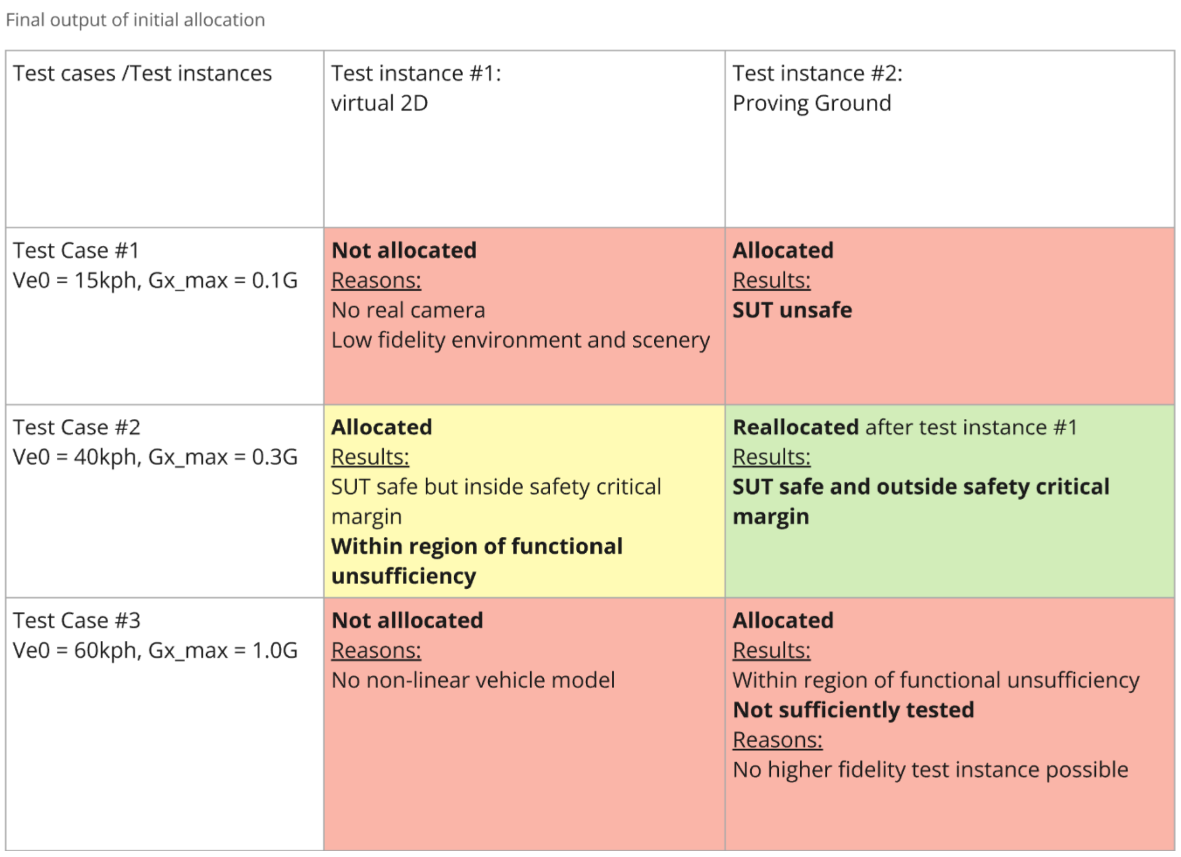
The proposed approach provides a framework for test allocation to test instances. However, special circumstances may allow deviating from the general methodology. Below, we outline a few counterexamples to the proposed process, showcasing special testing circumstances and their impact on allocation process:

1. <u>Availability of resources might increase selection of higher-fidelity test instances:</u> when resources are abundantly available, users may prefer testing on higher-fidelity test instances directly to achieve the highest reliability of results. In addition, the preparation time of a test instance, which otherwise has high throughput, may be a significant practical consideration for example, correctly configuring sensor models and vehicle dynamics for simulation or performing proving ground tests with the actual hardware.

2. <u>Multiple test cases could be combined and performed together, increasing effectiveness of test instances:</u> For example, in proving ground tests, a session starts and ends with the SUT at standstill. Therefore, certain types of testing could be combined together, for instance, test cases which evaluate system behaviour when starting from standstill with test cases which evaluate an interaction with another actor when the vehicle is at a non-zero vehicle speed. This (initial speeds zero) may also be the case for virtual or hybrid testing like DiL or HiL if specific initialisation (e.g., model settle times) or calibration phase is needed.

4.6 Documentation and results to be reported to SAF

First of all, the direct results of the initial allocation or decision-making process need to be reported to the “Execute” block of the SAF (see Figure 3) before being able to perform the according tests. These results are a kind of matrix or table with the test cases in one dimension and the allocated test instances as a second dimension (see a simplified example in Figure 32 based on the examples in Annexes 2 to 4). When a scenario cannot be allocated to any of the available test instances, it must be flagged as "not possible to execute" and reported to the “Coverage” block of the “Analyse” part of the SAF (see Figure 3). If further testing of a scenario was needed according to Figure 27 but a higher-fidelity test instance could not be allocated, the scenario must also be flagged, for example with a "not sufficiently tested" and reported to the “Coverage” block of the “Analyse” part of the SAF (as in Figure 3), together with details about the test instances where the scenario was executed and the reason due to which further testing was required.



Final output of initial allocation

Test cases /Test instances	Test instance #1: virtual 2D	Test instance #2: Proving Ground
Test Case #1 Ve0 = 15kph, Gx_max = 0.1G	**Not allocated** Reasons: No real camera Low fidelity environment and scenery	**Allocated** Results: **SUT unsafe**
Test Case #2 Ve0 = 40kph, Gx_max = 0.3G	**Allocated** Results: SUT safe but inside safety critical margin **Within region of functional unsufficiency**	**Reallocated** after test instance #1 Results: **SUT safe and outside safety critical margin**
Test Case #3 Ve0 = 60kph, Gx_max = 1.0G	**Not alllocated** Reasons: No non-linear vehicle model	**Allocated** Results: Within region of functional unsufficiency **Not sufficiently tested** Reasons: No higher fidelity test instance possible

Figure 28: Example initial allocation matrix based on examples from Annexes 2 to 4 (red colour: not allocated/not sufficiently tested or SUT unsafe, yellow colour: needs re-allocation, green colour: allocated and test results sufficient).

Besides the decision-making process as shown in Figure 27, more elaborate decision-making may be considered when to understand if scenarios must be further tested or reallocated to specific test instances after an initial allocation and test execution is performed (see D3.5). Such a process should build up on the previous initial allocation and not repeat some of the steps. To enable such a re-allocation (see D3.5) without any double work of the initial allocation, all the steps of the decision-making (reasons for decisions) should be documented and returned to SAF (as in Figure 34 and Annex 4). The reasons for decisions allow easier identification of potential test instances (e.g., from the comparison of test case requirements and test instance capabilities) or easier identification of missing or wrong metrics for the allocation. Finally, together with some validation or test results it supports the identification of missing or wrong test case requirements or test instance capabilities. For the documentation of the whole decision-making process of the allocation, a tree structure is recommended (mainly based on the structure from chapter 3 as in Annex 3, but extended by other metrics used for the decision-making mentioned in chapter 4.4) containing all metrics and results of the comparison to all test instances at once (feasible in digital format with options for zooming, filtering and grouping).

5 CONCLUSIONS

This document introduces a process for comparing the requirements of test cases with the capabilities of test instances. This process is divided into two parts. In the first part, the requirements of the test case are compared with the capabilities of the available test instances to see where the test cases could be executed. In the second part of the process, the decision-making with respect to certain metrics is done.

For the first part, a structure has been defined that is partly based on the ISO34503 standard [1] and expanded with additional elements (see chapter 3). The structure is divided into four main parts:

- **Scenery elements**: this part of the structure describes the physical environment, including details like road types, junctions, special structures, and temporary modifications. It helps define the context of the scenario and the capabilities needed from the test environment.
- **Environment conditions**: this part focuses on weather conditions, connectivity between actors, particulates, and illumination. These factors can significantly affect the performance of an ADS and need to be considered when choosing a test environment.
- **Dynamic elements**: this part deals with the moving objects within the scenario, including the subject vehicle, other traffic agents (vehicles, pedestrians, etc.), and human representation. It also explores additional elements like driving function, sensing capabilities, and vehicle dynamics.
- **Test Criteria**: this describes the additional criteria that are important but not part of a scenario description like evaluation criteria, completion criteria and capability criteria.

The second part that is described in this document is the decision-making process for allocating test cases to different test instances. The goal is to optimise the testing process by considering both technical and other metrics.

The process is based on two key elements (see chapter 4):

- **Metrics**:
    - Technical metrics assess the ability of a test instance to replicate the requirements of a test case. These include factors like criticality, sensor fidelity, and vehicle dynamics modelling.
    - Management metrics consider the cost and resources required to execute the test in different environments (simulation, XiL, proving grounds).
- **Prioritisation**:
    - Safety is the top priority.
    - The "virtual simulation first" approach is preferred whenever possible, to reduce costs and effort. However, limitations of simulation tools and the need for real-world validation might necessitate using other test instances.

    - Once a test is deemed insufficient in a lower-fidelity environment, it is iteratively allocated to the next cheapest higher-fidelity option until deemed sufficient or reaching the highest-fidelity option.

This approach ensures that testing is conducted efficiently while maintaining a high level of safety assurance.

The deliverable emphasises the importance of documenting the decision-making process for the Safety Argumentation Framework (SAF). This documentation helps the SAF understand the rationale behind test allocation decisions and potential limitations (see chapter 4.6).

The process described in this deliverable shall be used by the SUNRISE use cases in WP7 to initially allocate the test cases to their test instances. Some parts of the presented process include the reallocation of test scenarios to other test instances. This is an important part of the approach, but also overlaps with the work of Task 3.5 of SUNRISE "Validation Metrics for Test Runs and Dynamic Allocation". As this task finishes at the end of the project and will deal with the topic of reallocation in more depth, it is recommended to read deliverable 3.5.

While this document provides useful and structured information related to the allocation of test cases to test instances, it should not be observed as a full guide or instructions manual. For the comparison of test case requirements and test instance capabilities for example, it is not feasible to provide all details of an ODD structure, due to the wide variety of test instances and an even wider variety of test cases. Applying the contents of this deliverable may therefore require additional efforts or expertise to successfully allocate test cases to test instances.

6 REFERENCES

[1] *ISO 34503 - Road Vehicles - Test scenarios for automated driving systems - Specification for operational design domain [First edition 2023-08].*

[2] “https://openlabel.asam.net/V1-0-0/schema/openlabel_json_schema.json,” [Online].

[3] “https://openlabel.asam.net/V1-0-0/ontologies/openlabel_ontology_scenario_tags.ttl,” [Online].

[4] “BSI PAS 1883:2020 Operational Design Domain (ODD) taxonomy for an automated driving system (ADS) – Specification,” 2020. [Online].

[5] “https://www.asam.net/project-detail/asam-openodd-v100/,” [Online].

[6] F. Fahrenkrog, A. Fries, A. Das, S. Kalisvaart, E. Charoniti, O. Op den Camp and e. al, “Draft Prospective Safety Assessment Framework - Method,” *Milestone Report MS10 of the Horizon Europe Project V4SAFETY,* 2024.

[7] V4SAFETY Project, “Guidelines for the development, quality, and use of models of road-user behaviour and models for in-crash simulations in virtual safety assessment,” Project Deliverable No. D3.1, 2025.

[8] J. Klimke, D. Becker and L. Eckstein, “System Design of an Agent Model for the Closed-Loop Simulation of Relevant Scenarios in the Development of ADS,” in *29th Aachen Colloquium*, 2020.

[9] P. Ring, L. Wang, F. Fahrenkrog and O. Jung, “Modelling Cognitive Driver Behavior In The Context Of Prospective Safety Assessment,” in *26. Aachener Kolloquium*, Aachen, 2017.

[10] C. Siebke, M. Mai and G. Prokop, “What Do Traffic Simulations Have to Provide for Virtual Road Safety Assessment? Human Error Modeling in Traffic Simulations,” in *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[11] United Nations Economic Commission for Europe, *UN Regulation No. 157 – Automated Lane Keeping Systems (ALKS),* 2021.

[12] J. Scanlon, K. Kusano, J. Engström and T. Victor, “Collision Avoidance Effectiveness of an Automated Driving System Using a Human Driver Behavior Reference Model in Reconstructed Fatal Collision,” Waymo LLC, 2022.

[13] M. Steimle, T. Menzel and M. Maurer, “Toward a Consistent Taxonomy for Scenario-Based Development and Test Approaches for Automated Vehicles: A Proposal for a Structuring Framework, a Basic Vocabulary, and Its Application,” *IEEE Access,* vol. 9, pp. 147828-147854, 2021.

[14] ASAM, “ASAM OpenTestSpecification,” ASAM e.V., 2024.

[15] D. Fremont and al., “Formal scenario-based testing of autonomous vehicles: From simulation to the real world.,” in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020.

[16] G. E. Mullins, P. G. Stankiewicz and S. K. Gupta, “Automated generation of diverse and challenging scenarios for test and evaluation of autonomous vehicles,” *IEEE International Conference on Robotics and Automation (ICRA),* pp. 1443-1450, 2017.

[17] M. Winkelmann, C. Vasconi and S. Müller, “Transfer Importance Sampling - How Testing Automated Vehicles in Multiple Test Setups Helps With the Bias-Variance Tradeoff,” *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC),* pp. 26-31, 2022.

[18] A. Stocco, B. Pulfer and P. Tonella, “Mind the gap! a study on the transferability of virtual vs physical-world testing of autonomous driving systems,” in *IEEE Transactions on Software Engineering*, 2022.

[19] M. Steimle, N. Weber and M. Maurer, “Toward Generating Sufficiently Valid Test Case Results: A Method for Systematically Assigning Test Cases to Test Bench Configurations in a Scenario-Based Test Approach for Automated Vehicles,” *IEEE Access,* vol. 10, pp. pp. 6260-6285, 2022.

[20] HEADSTART, D4.2 - Evaluation results of application and demonstration, 2021.

[21] *ISO/ICE/IEE 29119-1:2022 Software and systems engineering - Software testing [Edition 2, 2022].*

[22] Z. Huang, H. Lam and D. Zhao, “Sequential experimentation to efficiently test automated vehicles,” *Winter Simulation Conference (WSC),* pp. 3078-3089, 2017.

ANNEX 1: RECOMMENDATIONS FOR THE COMPARISON OF TEST CASE REQUIREMENTS AND TEST INSTANCE CAPABILITIES

All standards stop at a specific level of detail to describe the ODD or a specific scenario description such as PAS1883 [4], ISO 34503 [1] or ASAM OpenSCENARIO. Otherwise, if all details would be described the rules would be no more applicable for each use case and could therefore be not declared as standard. For example one can look at precipitation where ASAM OpenSCENARIO v1.3.0 does not directly include the type hail (only rain or snow) and as value an intensity between 0 and 1, where it is on the simulation tool to interpret these values. Already rain can be described in several ways e.g., using terms such as low, medium or high or more detailed using physical quantities. Keep in mind that without any time frame the rain intensity cannot be described by the amount of rain expressed in volume per area which is typically part of weather data. There is also a strong relationship between rain intensity and wind direction and speed resulting in different raindrop shapes. There can be use cases, especially for function logic testing which needs a low degree of detail and perception tests which might need a high degree of detail for rain intensity. For this reason, a similar level of detail is taken for the structure in chapter 3 and the following recommendations as in the common standards.

Human-in-the-Loop

A purpose of the test case is to test realistically the Human-Machine-Interface (HMI) or in general the interaction of humans and the SUT results in high fidelity requirements until real human representation. From test instance capabilities as Simulator component either Human-in-the-Loop simulators (e.g., driving simulators or motorcycle riding simulators) or quasi-static ViL would match the requirement, as well as proving ground testing (including dynamic ViL). Proving ground testing is also covered by the Simulator component because a specific situation or scenario is provoked which is a kind of simulation, but no virtual simulation in that sense. More details can be specified for the test case requirements or test instance capabilities, such as details for the visualization and HMI, as well as motion cuing in case of dynamic simulators. This includes details on the fidelity of the virtual environment, such as the need for 2D or 3D, and the level of photo-realism. The available interfaces for HMI grading, such as screens, AR glasses, speakers, touch screens, microphones, pedals, and buttons can also be specified.

As soon as there is a need for Human-in-the-Loop the real-time requirement should be stated in the test criteria. If the test instance is slower or faster than real-time it should not be allocated for such test cases requiring Human-in-the-Loop.

Communication Interface

The communication interface covers Vehicle-to-Everything (V2X) communication also other communication types such as Bicyclist-to-Everything (B2X), which may have specific requirements on the message or data content (e.g., positions, time, response, warnings) or format (like ETSI ITS CAM, DENM, SPATEM).

Implementation effort of virtual components

In general, whenever software components are integrated into a test instance there can be several issues resulting from differences between system development and its testing, like specific software or software versions, compiler issues (e.g., different operating systems, libraries) or hardware requirements. This means among test case requirements there can be specific SUT requirements (e.g., software or software versions) which should also be compared to the test instance capabilities to estimate the implementation effort.

Safety hazard metric

Different metrics can be used to conclude on the safety hazard of a test case. First of all, there might be the source of the test case in the test criteria, such as police reported data or in-depth crash database partly containing more details of collision speeds, crash severity or injury levels. Alternatively, any kind of risk assessment data could be part of the test criteria. Second, the test case itself may contain intersections or in general crossing trajectories such as cut-in scenario. Finally, the status of the SUT can be used to derive the safety hazard, means the degree of automation or control in combination with its degree of maturity (e.g., release stages from alpha to gamma) or in similar way the time range to rollout. It is recommended to follow the ISO standards related to safety such as Functional Safety (FuSa) ISO26262 or Safety Of The Intended Function (SOTIF) ISO21448.

Safety hazard on proving grounds

Safety hazardous test cases do not automatically mean an exclusion of proving ground allocation. Different proving ground testing solutions exist, with robots as targets (see Figure 33) or remote or automated control of vehicles to prevent safety hazards or for risk reduction. The usage of safety drivers as backup solutions for ADAS or ADS malfunctions (in the car or via remote) is also a countermeasure for potential risks (see Figure 34). Another opportunity for real tests is the usage of rovers (see HEADSTART Deliverable 4.2 [20]). Such setups for proving ground testing based on robots or remote control have a much higher safety hazard margin compared to proving ground tests with direct involvement of humans (e.g., human drivers). Therefore, it is recommended to take all possible proving ground testing setups into account (all proving ground setups’ capabilities) to find an appropriate real testing solution for the test case execution.

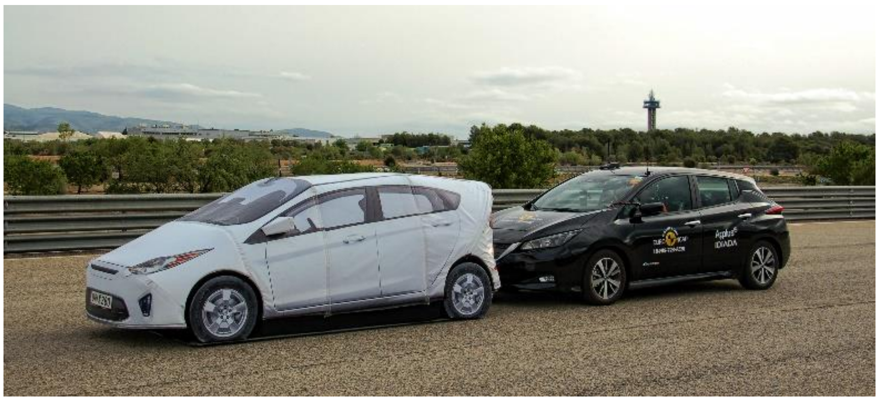


Figure 29: Euro NCAP AEB test on proving ground (potential collision of real drivers with target, source: Applus IDIADA)

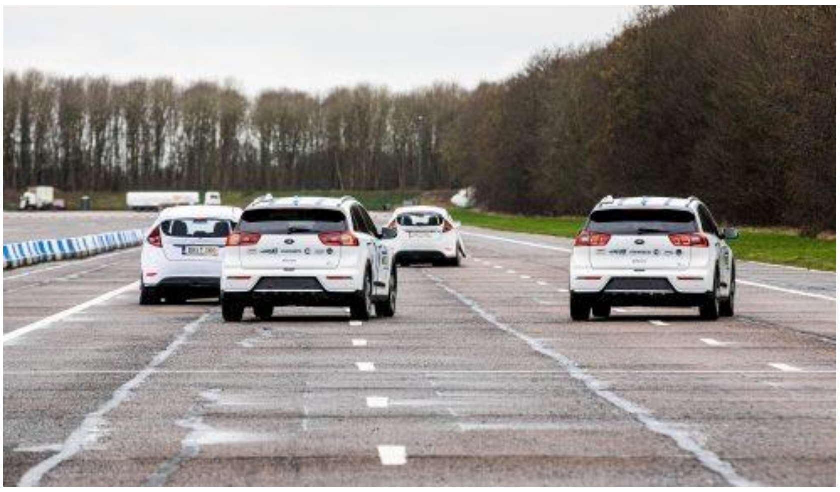


Figure 30: Proving ground testing with safety drivers as backup solution in case of ADS faults to prevent collisions (source: MuCCA project)

Metrics' lower and upper thresholds

Keep track on lower and upper thresholds of the metrics for the comparison of test case requirements and test instance capabilities. This means the test instance capabilities should lie within the ranges of the test case requirements. For example, if the test case (with the purpose of functional testing) requires a simplified vehicle dynamics model (for smooth trajectories), several models could be used such as single-track or two-track models. However a too realistic MBS model could maybe not be parameterised and therefore not be used for the test case. The lower threshold of the test case could be a single-track model which would exclude e.g. point-mass models and the upper threshold a two-track model which would exclude e.g. MBS models.

ANNEX 2: EXAMPLES OF TEST CASE REQUIREMENTS STRUCTURES

Let us pick three test cases from the EU Regulation No.157 from the deceleration scenarios (see Figure 31).

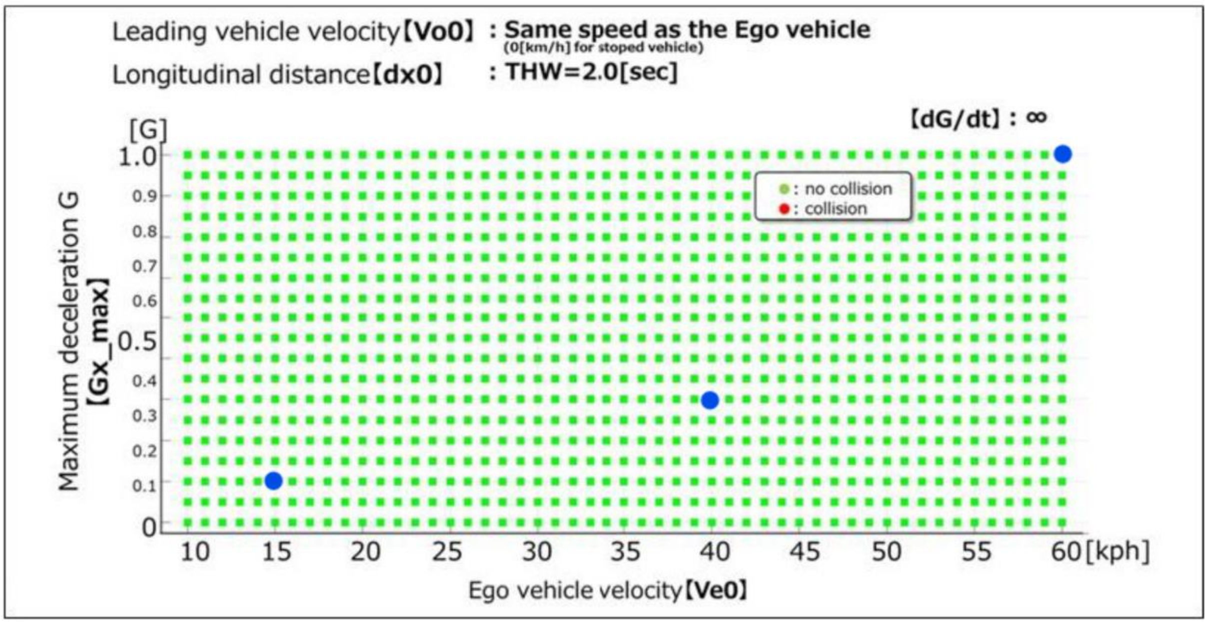


Figure 31: Picked Deceleration test cases from EU Regulation No. 157 (ALKS)

Just for demonstration we define here completely different test purposes (for test case #3 the test purpose is not specified to not influence the structure and finally the allocation) for the three different test cases and show the differences in the ODD structure. In practice, test cases with such different test purposes would not be allocated at once and would not appear in the same output matrix of the allocation process.

To keep it as simple as possible not the whole structure of chapter 3 is defined, only the most important elements for such an allocation and also no requirements on the road such as a straight road of a specific length. In practice according to chapter 3.3 it is not recommended to omit these elements.

The first test case is at low speed with low decelerations which would result in few test case requirements. To have some limiting test case requirements this test case was chosen for the test of the sensing and perception system (sensor performance under rain conditions). This test purpose has a strong influence on the sensor fidelity (degradation caused by rain and reflections by wet surfaces), as well as scenery (wet road surfaces) and environment (rain as precipitation) elements' fidelity as shown in Figure 32.

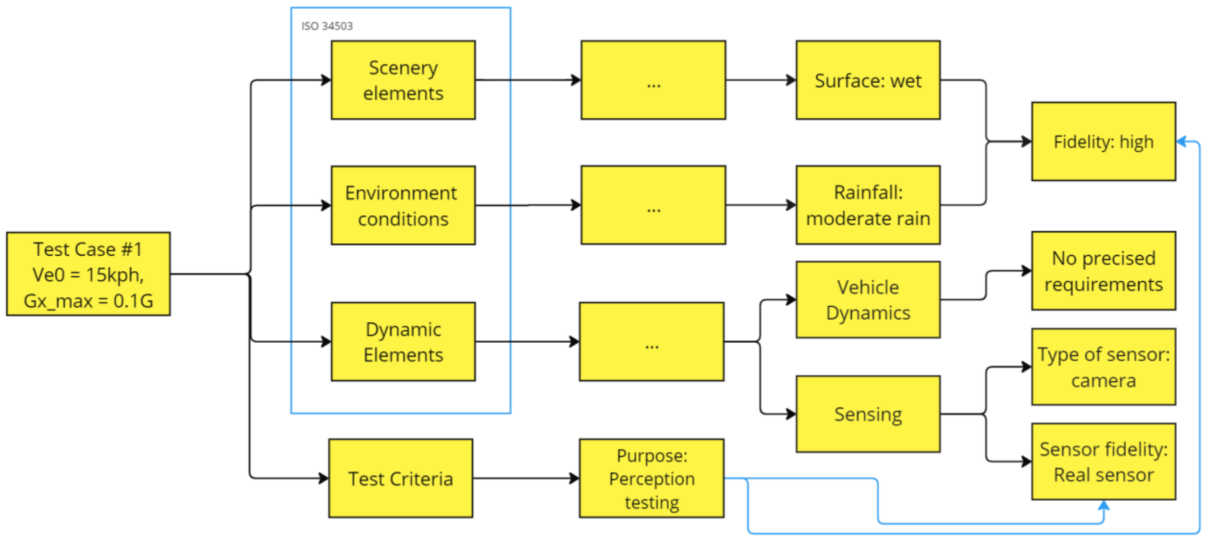


Figure 32: Structured test case requirements for test case #1

The second test case (Figure 37) is again at low speed with small deceleration values and is finally very similar to test case one. But here a contrary test purpose was chosen, which is the function-logic test or decision-making test (here: low or no precise requirements in general). To make the influence of the test purpose as one part of the test criteria more visual it was decided to precise the minimal requirements of the sensing part, another opportunity would be to have a graph from dynamic elements to “No precise requirements”.

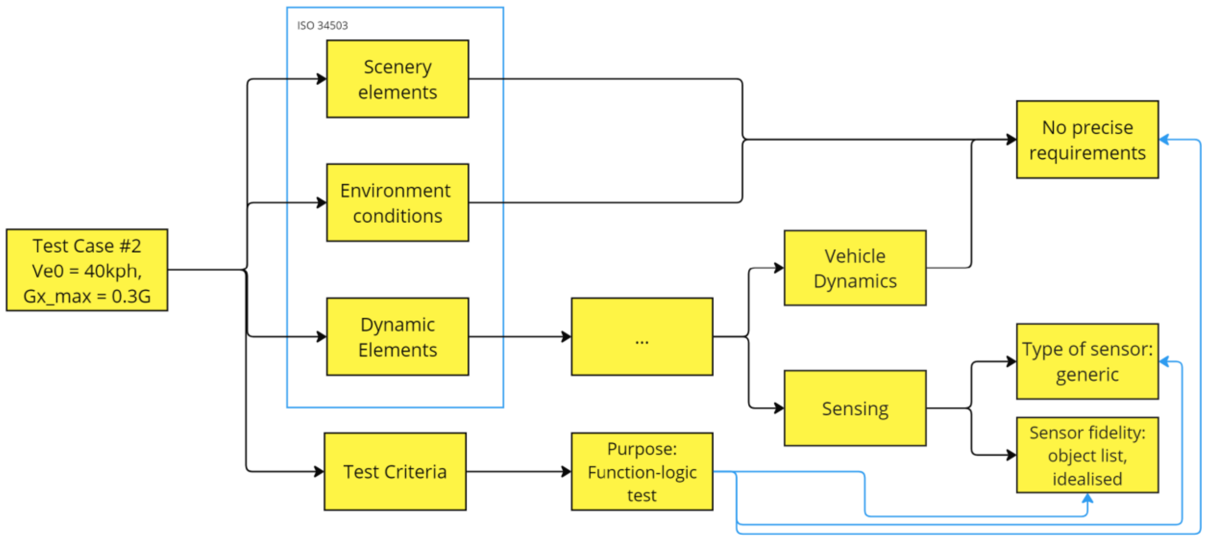


Figure 33: Structured test case requirements for test case #2

The third and last test case (Figure 38) for the demonstration is the one in upper right corner of the Figure 31. Since it is at a higher speed with high deceleration values there are some specific requirements from the scenario parameters such as the bad fidelity of linearized vehicle models (requirement for exclusion of linearized vehicle models). Therefore, no additional test criteria were precise compared to the first two test cases.

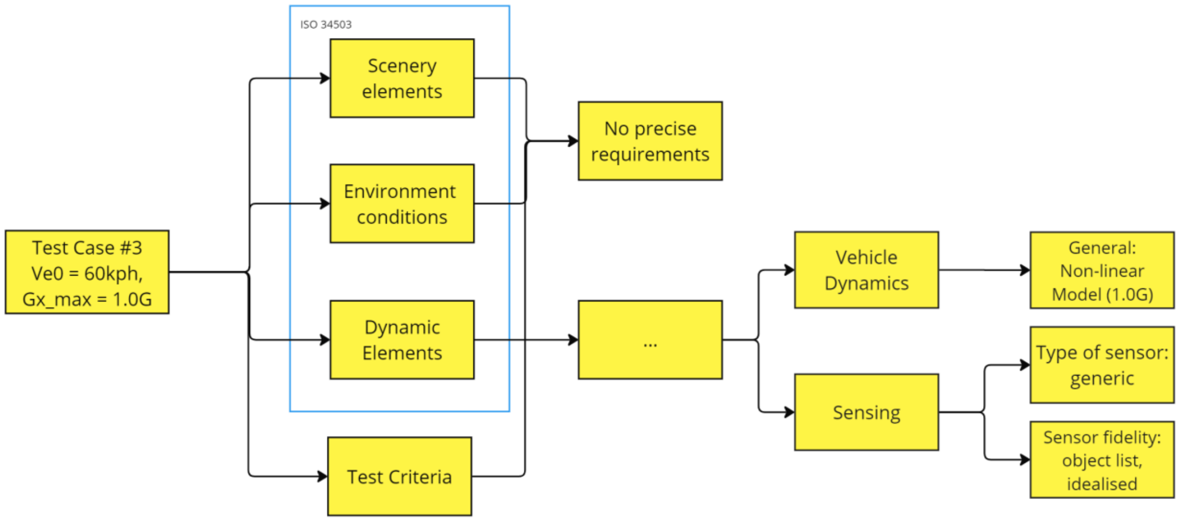


Figure 34: Structured test case requirements for test case #3

ANNEX 3: EXAMPLES OF TEST INSTANCE CAPABILITIES STRUCTURES

For instance, as given test instances a virtual 2D simulation platform was chosen and proving ground testing. To keep the structures simple for a comparison to the test case requirements in Annex 2 it was only concentrated on the most important elements for the comparison to the test case requirements from Annex 2 or to see at first glance the differences between both test instances e.g., the purpose of a test instance. In practice, according to chapter 3.3 it is not recommended to omit these elements.

The first test instance is a virtual simulation platform (Figure 35) where only linearised single-track models are available and generic object list-based sensor models which are idealised (Ground-Truth information). The environment and scenery are two-dimensional and very simplified (e.g., no weather conditions or effects).

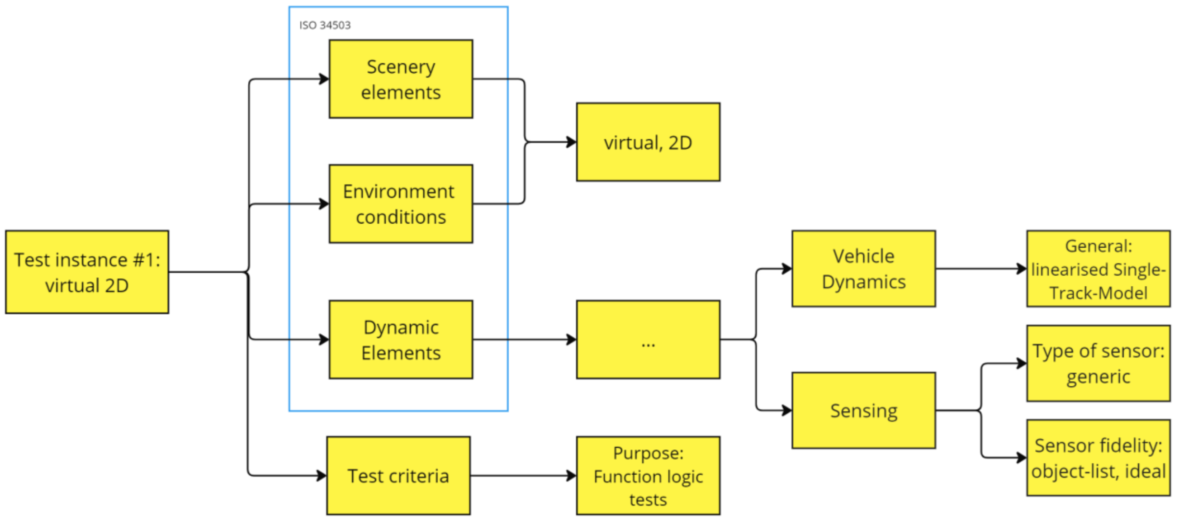


Figure 35: Structured test instance capabilities for test instance #1

The second test instance is a proving ground. Since the requirements on the road (straight road of a specific length) are quite low it is assumed that these are always fulfilled and not explicitly mentioned here as test instance capabilities. All the elements on the proving ground shall be real, which means real vehicle, real sensors, and further (Figure 36).

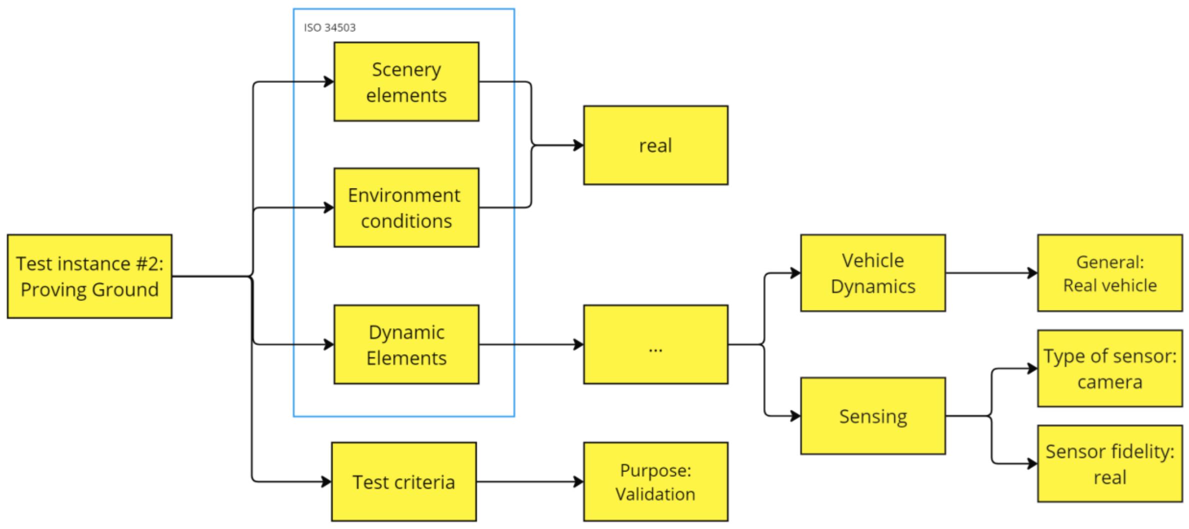


Figure 36: Structured test instance capabilities for test instance #2

ANNEX 4: EXAMPLES OF STRUCTURE COMPARISONS OF TEST CASE REQUIREMENTS AND TEST INSTANCE CAPABILITIES

Here the focus is set on the test cases and not on the test instances. Per the test case from Annex 2 the test instances from Annex 3 are compared to for the allocation.

Test case #1 from Annex 2 has strong requirements for the fidelity of the scenery, environment and sensor models from dynamic elements. Only test instance #2 has these capabilities (Figure 37) and test instance #1 has none of these capabilities, see Annex 3, Only regarding the vehicle model test case #1 has no requirements precise which is fulfilled by both test instances.

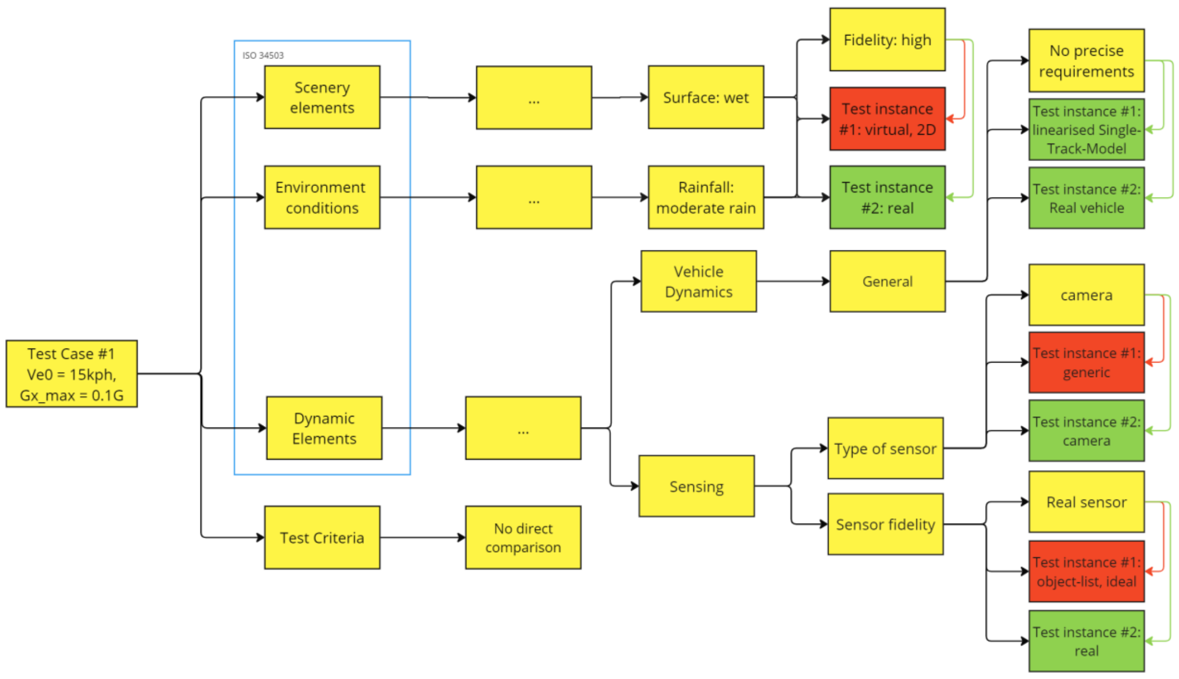


Figure 37: Structured comparison of requirements of test case #1 to test instance capabilities of test instances #1 and #2

As visualised in Figure 38 test case #2 from Annex 2 has very low requirements everywhere which can be fulfilled by both test instances from Annex 3.

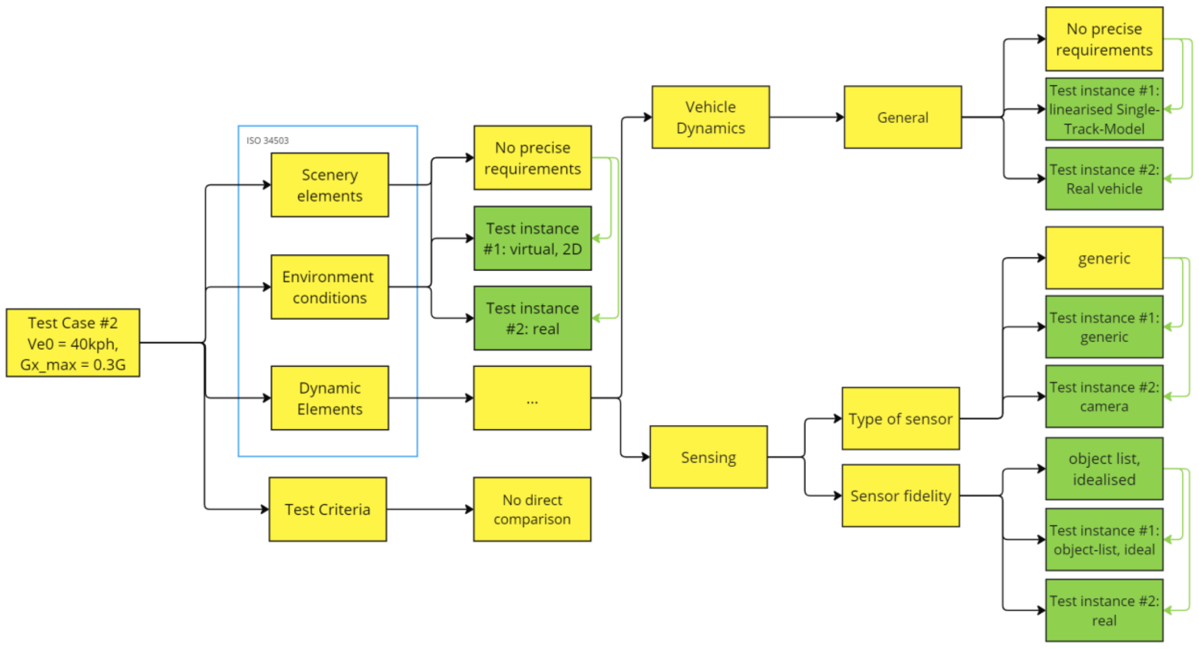


Figure 38: Structured comparison of requirements of test case #2 to test instance capabilities of test instances #1 and #2

Additionally, this test case is used as a show case how the metrics are applied with concrete values in order to generate an ordered list of test instances according to formula (1) or (2) from chapter 4.3. Two metrics are taken, the first one is the binary sensor fidelity metric. Since both test instances fulfill the test case requirements for sensor fidelity the value of this metric is 1 for both test instances. A second metric the test throughput is taken as a continuous metric to get an ordered list (decreasing metric values) with the most efficient test instance first. It is assumed to have a test throughput of the virtual test instance #1 of 240 test cases per day if each takes 10 seconds and a test throughput of 2.4 test cases per day for test instance #2 if each test case takes 10 seconds. Since normalised metrics are needed the test throughput is divided by the highest value, which is 240, therefore test instance #1 has a test throughput of 1 and test instance #2 of 0.01. Both metrics are weighted by 0.5. Applying formula (1) or (2) one gets for test instance #1 a total metric value of 0.5 * 1 + 0.5 * 1 = 1 and for test instance #2 0.5 * 1 + 0.5 * 0.01 = 0.5005. The list of test instances with decreasing order of metric values starts with the most efficient test instance #1 (metric value of 1) which is followed by test instance #2 with a total metric score of 0.5005.

As visualised in Figure 39 test case #3 from Annex 2 has low requirements nearly everywhere except for the exclusion of linearised vehicle models which cannot be fulfilled by test instance #1 from Annex 3.

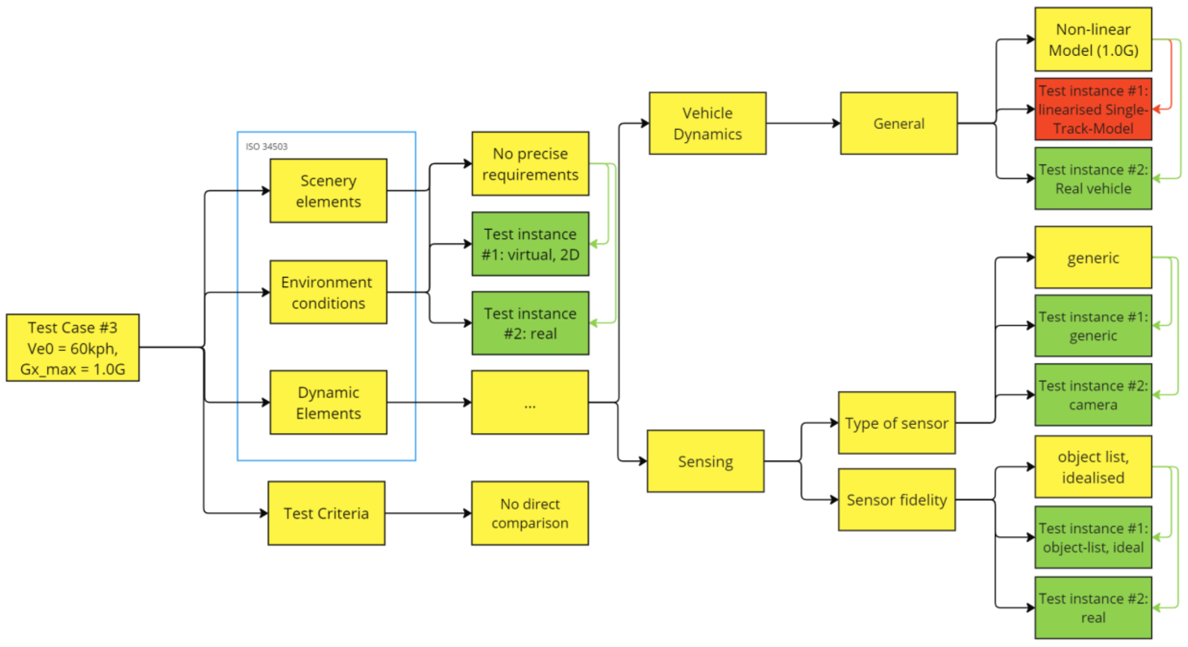
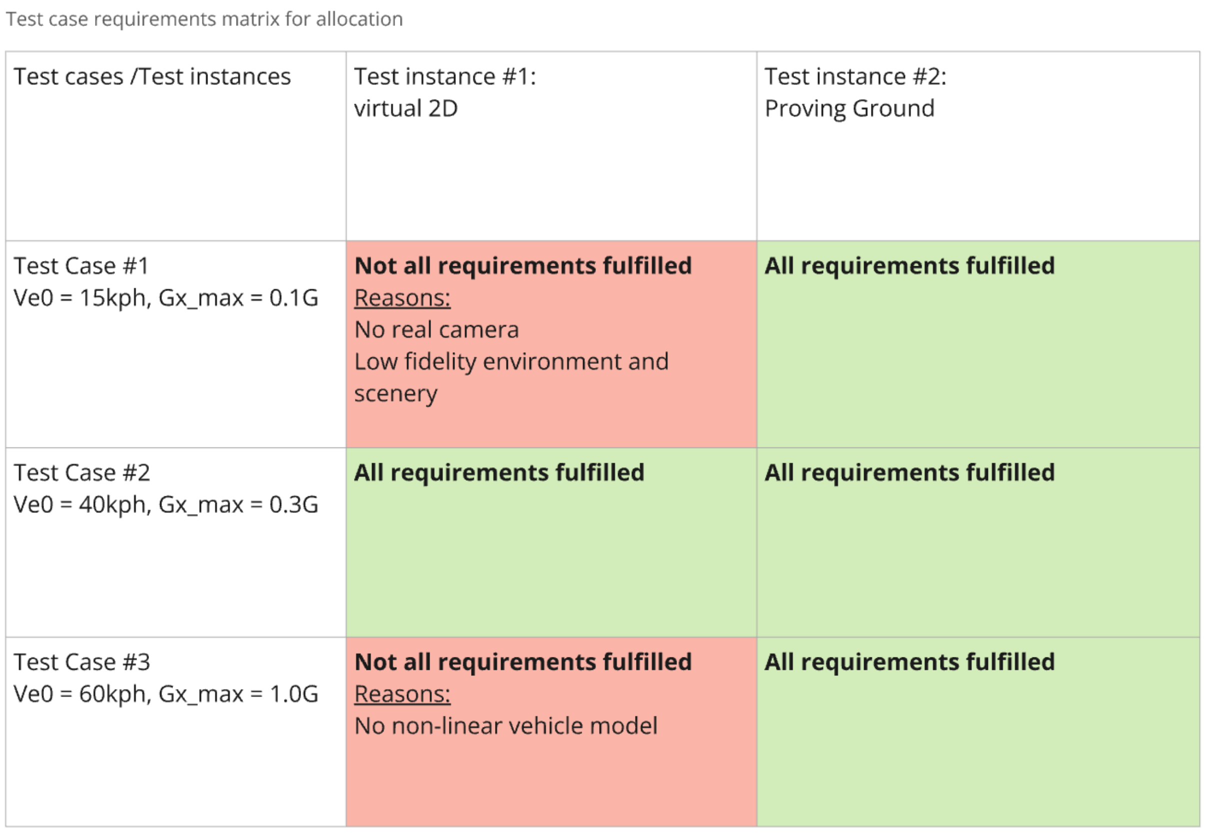


Figure 39: Structured comparison of requirements of test case #3 to test instance capabilities of test instances #1 and #2

As a summary or overview, a matrix was created summarizing the most important information of the comparison of test case requirements to test instance capabilities for the allocation (Figure 40)


Test case requirements matrix for allocation

Test cases /Test instances	Test instance #1: virtual 2D	Test instance #2: Proving Ground
Test Case #1 Ve0 = 15kph, Gx_max = 0.1G	**Not all requirements fulfilled** Reasons: No real camera Low fidelity environment and scenery	**All requirements fulfilled**
Test Case #2 Ve0 = 40kph, Gx_max = 0.3G	**All requirements fulfilled**	**All requirements fulfilled**
Test Case #3 Ve0 = 60kph, Gx_max = 1.0G	**Not all requirements fulfilled** Reasons: No non-linear vehicle model	**All requirements fulfilled**

Figure 40: Comparison matrix of test cases requirements and test instances capabilities for allocation

ANNEX 5: EXAMPLE OF THE STRUCTURE FOR SUB-UC 1.3

This chapter outlines the designed sub-UC 1.3 allocation structure, using an example of simulating the exchange of collective perception messages (CPMs) as detailed in deliverable D7.1. Out of the three test scenarios, two are described as follows:

- sub-UC 1.3-A: "Darting-out pedestrian", which is based on simulation with a perspective to make it hybrid
- sub-UC 1.3-B: “Urban junction”, which is based on hybrid simulation

In this subchapter, those two test scenarios (sub-UCs 1.3-A & 1.3-B) are briefly described and visualised via a sketch or a simulation environment snapshot. Then, the building blocks representing the test instance capabilities are displayed as a tree structure (see Figure 42 and Figure 44) following the template structure of chapter 3.3. In there, a colour notation denotes the test case components’ allocation to different test environments, namely XiL, Simulated, Real-world or assigned to the category “Not considered”.

<u>**sub-UC 1.3-A: "Darting-out pedestrian"**</u>

The scenario for sub-UC 1.3-A involves a VRU (pedestrian) and two CAVs (ego & CAV #02). In Figure 41, a snapshot from a scenario running in a co-simulation environment including SCENIC, CARLA and ROS modules is presented.

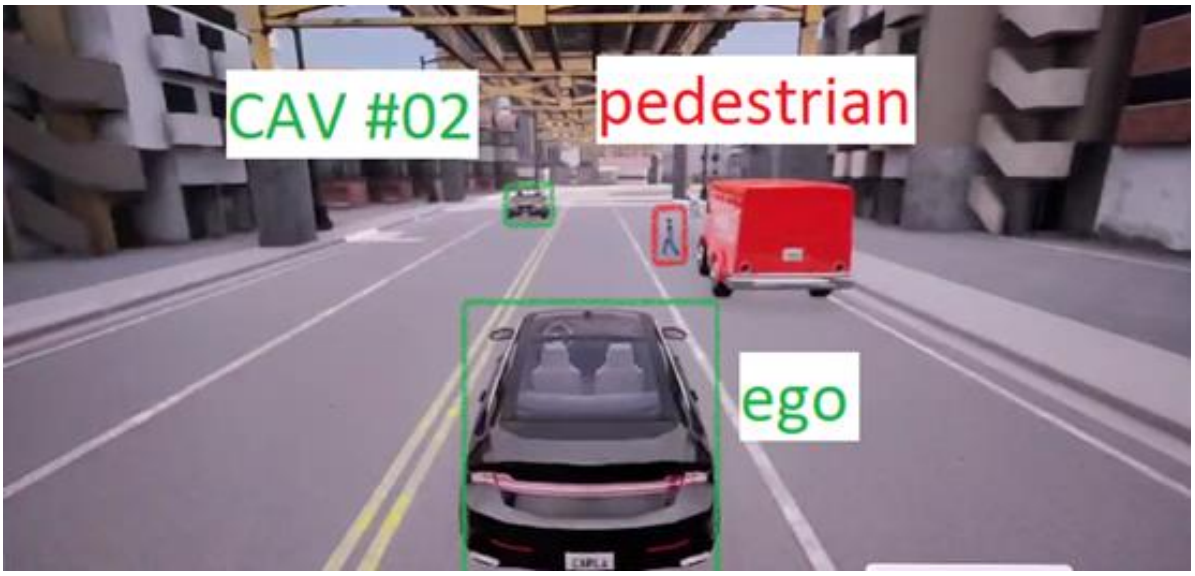


Figure 41: Schematic representation of sub-UC 1.3-A “darting-out pedestrian” test scenario

On a straight road chapter, a connected ego vehicle moves forward and another connected one (CAV #02) approaches from the opposite direction. In front of the ego vehicle on the right side of the road a heavy vehicle (truck or bus, denoted here in red colour) is parked, blocking the ego’s line of sight (field of view) for a pedestrian who is occluded behind the truck and who intends to cross the road.

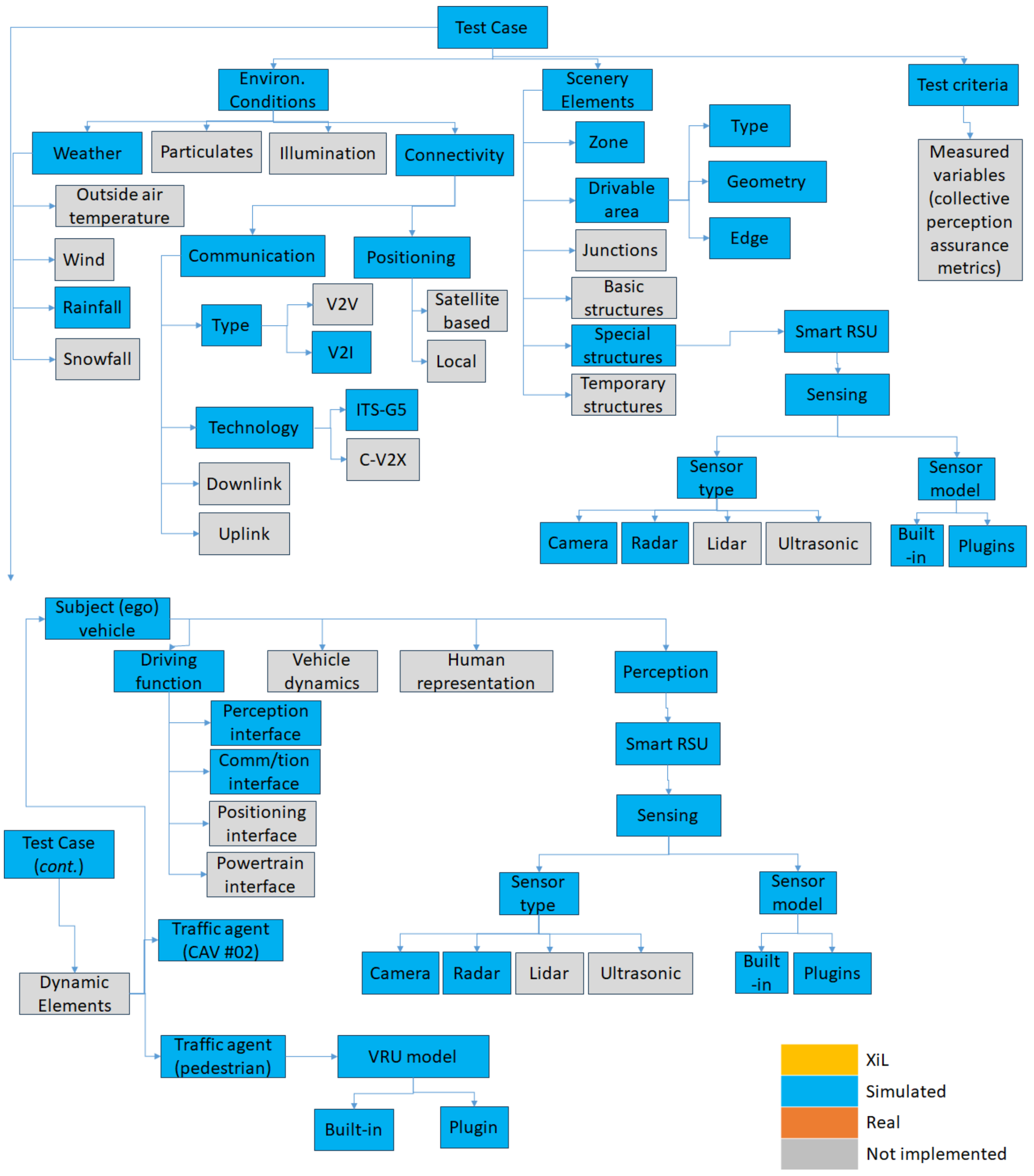


Figure 42: Test instance structure of sub-UC 1.3-A "darting-out pedestrian"

The action of crossing the road from the pedestrian is perceived by the opposite approaching vehicle (CAV #02), which informs the ego vehicle via V2V communication (through the exchange of V2I and I2V messages). This way the ego vehicle's perception functions collectively, exploiting fusion techniques that improve the original disadvantageous perception status. The expected response is that the ego vehicle decelerates or proceeds to a safe manoeuvre/stop.

Figure 42 shows the tree diagram, which structures the simulation components of the sub-UC 1.3-A test scenario, highlighting simulated and not implemented blocks (XiL and real are not displayed).

The main environmental condition for this instance is connectivity. The preferred simulated communication mode is V2I (and I2V) between the two CAVs (ego & #02). Regarding the scenery elements, the straight road chapter is simulated including the edge and any available curb space. The ego vehicle’s perception is based on the sensor types of camera and radar, using high-level fusion. The CAV #02 uses the same simulation pattern, while the other main traffic agent (pedestrian) is also simulated through CARLA software.

sub-UC 1.3-B: "Urban junction"

The scenario for sub-UC 1.3-B is designed to simulate Vulnerable Road Users (VRUs) and Connected Autonomous Vehicles (CAVs) interacting at urban intersections (see Figure 43). In sub-UC 1.3-B, the CAVs receive aggregated cooperative sensing information from Mobile Edge Computing (MEC) servers, allowing them to make informed decisions to ensure safety and avoid collisions.

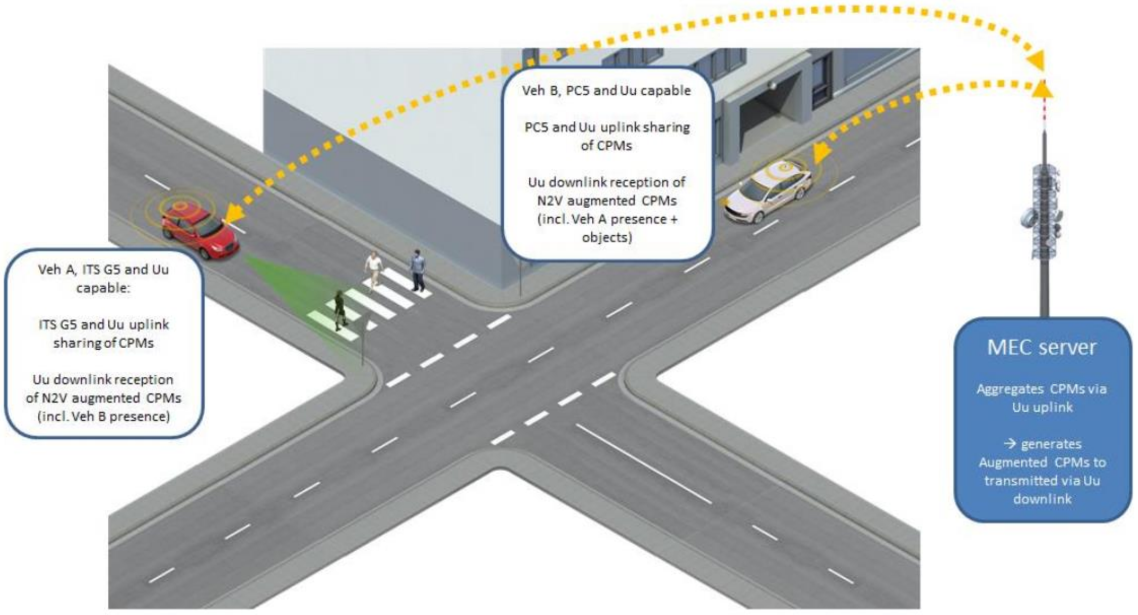

Veh B, PC5 and Uu capable
PC5 and Uu uplink sharing of CPMs
Uu downlink reception of N2V augmented CPMs (incl. Veh A presence + objects)
Veh A, ITS G5 and Uu capable:
ITS G5 and Uu uplink sharing of CPMs
Uu downlink reception of N2V augmented CPMs (incl. Veh B presence)
MEC server
Aggregates CPMs via Uu uplink
→ generates Augmented CPMs to transmitted via Uu downlink

Figure 43: Schematic representation of sub-UC 1.3-B “urban junction” test scenario (image source: [47] in D7.1).

The following sub-cases are considered within sub-UC 1.3-B:

Test sub-case #1: CAV Planning to Cross Straight Ahead

- Description: The CAV (ego vehicle B) intends to proceed straight at an intersection.
- Trigger: The CAV receives information indicating another vehicle (vehicle A) is also crossing straight with dynamics suggesting a potential collision.
- Response: The CAV slows down and possibly stops to prevent a collision.

Test sub-case #2: CAV Planning to Turn Left

- Description: The CAV (ego vehicle B) intends to turn left at an intersection.
- Trigger: The CAV receives information indicating vehicle A is blocking its route, posing a potential collision risk.
- Response: The CAV (ego vehicle B) slows down smoothly and waits for vehicle A to clear the intersection before proceeding.

Test sub-case #3: CAV Planning to Turn Right

- Description: CAV intends to turn right at an intersection.
- Trigger: Vehicle A perceives the pedestrians and send this information to the MEC server. CAV receives information from MEC server indicating multiple pedestrians are crossing the street along its route, posing a potential collision risk.
- Response: CAV slows down smoothly and waits for the pedestrians to cross before proceeding into the intersection conflict area.

To effectively simulate the test cases outlined above, a detailed description of the simulation components and their relationships is essential. It is necessary to define whether these components are simulated, Hardware in the Loop (HiL), or real. The scenario allocation structure, presented as a tree diagram in Figure 44, outlines the various components required for simulating the specified test cases within sub-UC 1.3-B.

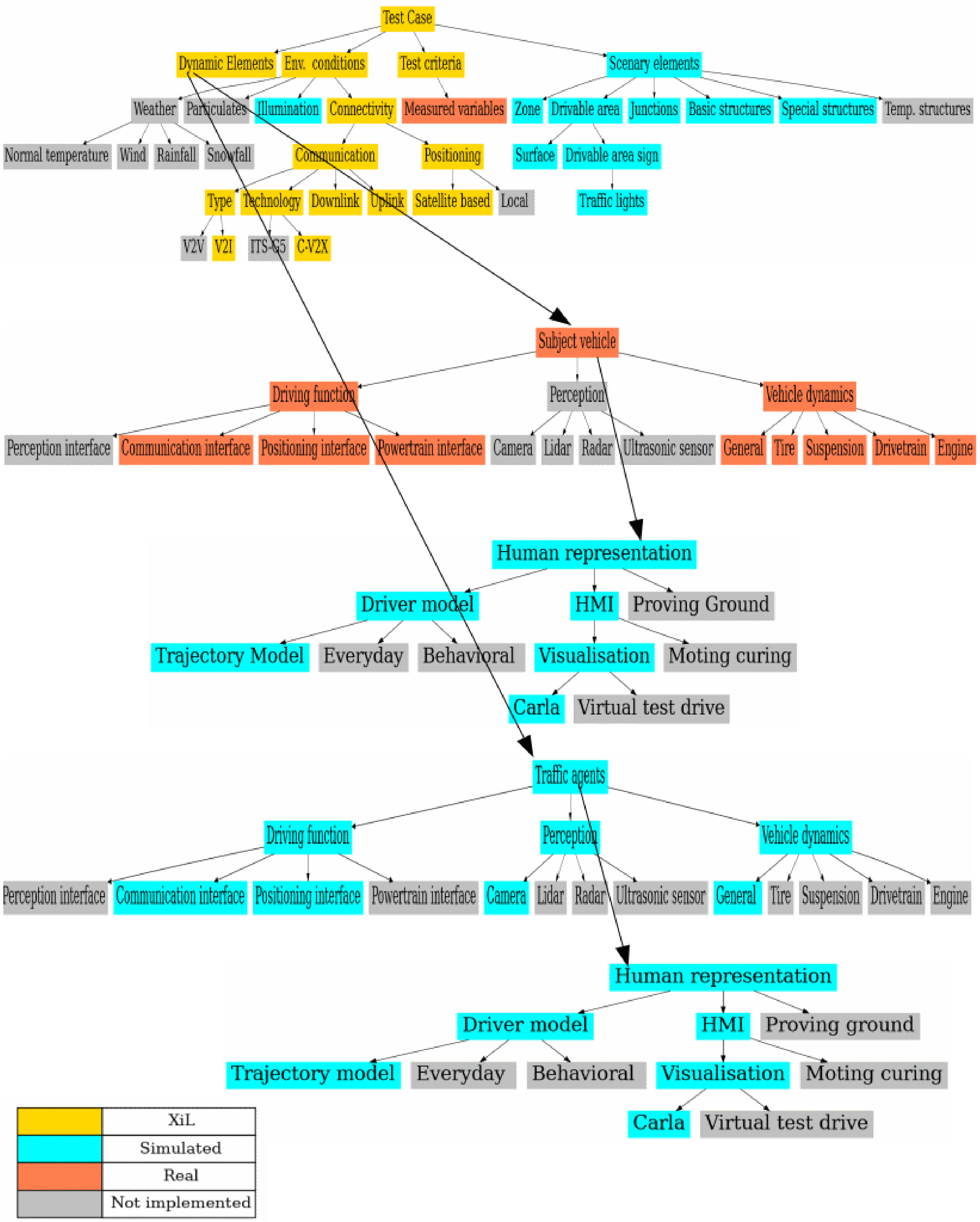


Figure 44: Test instance structure of sub-UC 1.3-B "urban junction"

The structure obtained for sub-UC 1.3-B test instance, as illustrated in Figure 44, presents a hierarchical depiction of simulation components. The nodes within the tree diagram are color-coded to signify their nature: XiL components are depicted in gold, simulated components in blue, real components in coral, and unimplemented components in grey.

There exists a hierarchical relationship between parent nodes and their respective children nodes within the structure. If the parent node is categorized as XiL, then either at least one child node is XiL, or the children nodes comprise a mixture of simulated and real components, or the children nodes are marked as not implemented. Conversely, if the parent node is simulated, then all of its implemented children nodes are simulated. Similarly, if the parent node is categorized as real, then all of its implemented children nodes are real.

At the root of the structure lies the test case instance, functioning primarily as a HiL or XiL test case. It branches into four children nodes: XiL dynamic elements, XiL environmental conditions, XiL test criteria, and simulated scenery elements.

Dynamic elements within this test case instance predominantly operate within a XiL environment. They include a real subject vehicle and simulated traffic agents, such as vehicle A and pedestrians. The subject (or ego) vehicle is equipped with genuine driving, communication, positioning, and powertrain interfaces, alongside authentic vehicle dynamics components. However, human representation is simulated using drive and trajectory models. Visualization is facilitated through simulation visualization using Carla.

Traffic agents, excluding the subject (or ego) vehicle, are entirely simulated. They feature driving, communication, and positioning interfaces, with perception simulated via a camera model. Simulated general vehicle dynamics represent their vehicle dynamics, while human representation mirrors that of the subject vehicle.

Environmental conditions primarily operate within a XiL environment, with simulated illumination and connectivity. Communication, particularly V2I communication employing Cellular-vehicle-to-everything (CV2X) technology, is also XiL. Positioning relies on satellite-based systems, operating within an XiL environment.

Test criteria are assessed using real variables to ensure a precise evaluation of system performance. On the other hand, scenery elements are entirely simulated, creating a lifelike environment for scenario testing.