Methodology for Test Case Allocation based on a Formalized ODD

Martin Skoglund$^{1[[0000-0001-6901-4986]}$, Fredrik Warg$^{1[0000-0003-4069-6252]}$, Anders Thorsén$^{1[0000-0001-7933-3729]}$, Sasikumar Punnekkat$^{2[0000-0001-5269-3900]}$, and Hans Hansson$^{2[0000-0002-7235-6888]}$

^{1} Department of Electronics, RISE Research Institutes of Sweden, Borås, Sweden
martin.skoglund@ri.se,fredrik.warg@ri.se,anders.thorsen@ri.se
^{2} MRTC, Mälardalen University, Västerås, Sweden
sasikumar.punnekkat@mdu.se,hans.hansson@mdu.se

Abstract. The emergence of Connected, Cooperative, and Automated Mobility (CCAM) systems has significantly transformed the safety assessment landscape. Because they integrate automated vehicle functions beyond those managed by a human driver, new methods are required to evaluate their safety. Approaches that compile evidence from multiple test environments have been proposed for type-approval and similar evaluations, emphasizing scenario coverage within the system's Operational Design Domain (ODD). However, aligning diverse test environment requirements with distinct testing capabilities remains challenging.
This paper presents a method for evaluating the suitability of test case allocation to various test environments by drawing on and extending an existing ODD formalization with key testing attributes. The resulting construct integrates ODD parameters and additional test attributes to capture a given test environment's relevant capabilities. This approach supports automatic suitability evaluation and is demonstrated through a case study on an automated reversing truck function. The system's implementation fidelity is tied to ODD parameters, facilitating automated test case allocation based on each environment's capacity for object-detection sensor assessment.

Keywords: Safety assurance · Operational design domain · Automated systems · Test case allocation.

1 Introduction

The safety assurance of Connected, Cooperative, and Automated Mobility (CCAM) systems is a critical challenge for their widespread adoption. As higher levels of automation are pursued, traditional validation through real-world testing becomes impractical due to the immense number of scenarios required. In the automotive field, this is commonly called the "billion-miles" challenge [12] but extends to any domain with automation ambitions. An appropriate mix of physical and virtual testing has emerged as a more feasible solution in such contexts. A blended physical and virtual strategy is, therefore, the practical alternative.

Despite these efforts, a significant gap remains between high-level schematic descriptions and practical guidance in concrete methods. The lack of a practical validation hampers the safe and large-scale deployment of CCAM technologies, with many still under development or recently introduced.

Today, scenario-based testing for automated driving is growing in importance and prevalence. However, it is still a challenge to determine if a test suite sufficiently covers the ODD [25]. Part of solving this is to develop systematic methods to align scenario requirements with distinct test environment capabilities. Integrating ODD parameters with test attributes can address this gap by enabling automated test case allocation to appropriate test environments. A subcategory of this topic is an external assessment of the appropriateness of such allocations, as required by functional safety standards [7]. This assessment is similarly complex for the reasons that hamper the initial allocation, particularly scope, and link to intended context and test environment appropriateness [17].

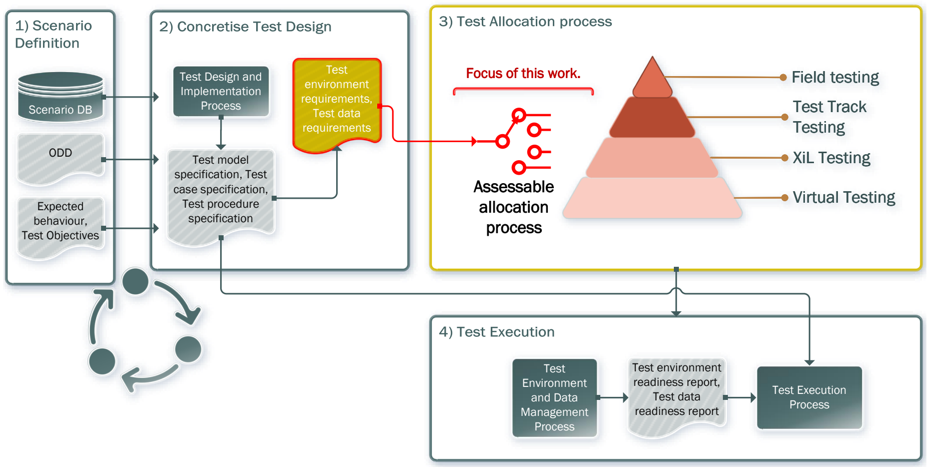
1) Scenario Definition
Scenario DB
ODD
Expected behaviour, Test Objectives
2) Concretise Test Design
Test Design and Implementation Process
Test environment requirements, Test data requirements
Test model specification, Test case specification, Test procedure specification
3) Test Allocation process
Focus of this work.
Assessable allocation process
Field testing
Test Track Testing
XiL Testing
Virtual Testing
4) Test Execution
Test Environment and Data Management Process
Test environment readiness report, Test data readiness report
Test Execution Process

Fig. 1: Schematic overview of a scenario-based safety-assurance process highlighting the allocation step.

Building on Road vehicles - Test scenarios for automated driving systems ISO 3450X [8–10] and an ODD formalization [18], this paper proposes an automated test case allocation process centered on extending an ODD object with test environment attributes. The ODD parameters alone do not fully reflect a test environment's capacity to address hazards, complexity, and fidelity. Accordingly, we extend the ODD concept with additional test environment attributes, forming a unified structure that better aligns test scenario requirements with test environment capabilities.

The approach confines test requirements to specified capabilities, allowing for the evaluation and automated allocation of test cases based on each environment's capacity to provide relevant safety evidence. Cost and scheduling considerations, which lie outside the scope of functional safety, are excluded to maintain focus on safety-specific concerns.

This study builds on the ISO 3450x scenario framework and a recent formalization of operational design domains to propose an automated allocation method that augments the ODD with test-environment descriptors [16]. Confining test requirements to declared capabilities enables objective allocation of scenarios to those environments that can produce credible safety evidence; this claim is illustrated with an automatic reversing-truck case study supported by an open-source implementation. The remainder of this paper is organized as follows. Section 2 reviews related work on scenario-based safety assurance. Section 3 details the proposed methodology for test allocation based on a formalized ODD. Section 4 presents the reversing-truck case study, and Section 5 summarises the principal findings and outlines avenues for future research.

2 Background and Related Work

Automated driving functions exemplify the broad challenges associated with CCAM safety assurance. Growing complexity and variant diversity lead to exponentially increasing testing demands that exceed the capacity of conventional requirement-based approaches. Many initiatives adopt a scenario-based perspective to address the increased complexity, at least for top-level testing [8–10]. This practice can improve coverage of diverse and potentially unforeseen corner cases while enabling reuse across different functionalities. However, it also creates challenges in ensuring completeness. Scenario-based tests often require significant computational and organizational resources for test design, execution, and assessment across heterogeneous environments, and the principal challenge lies in implementing these methods at scale [15]. The approach offers increased flexibility in adapting to evolving test requirements by decoupling scenarios from complex, difficult-to-maintain test code. Its practical relevance is underscored by UNECE Regulation No. 157, which governs automated lane-keeping systems and highlights the importance of scenario-based testing in ensuring robust system performance [24]. Scenario-based safety assurance approaches can be seen as an extension of the dynamic testing described in Software and systems engineering — Software testing ISO 29119 [11], which more comprehensively addresses processes, documentation, techniques, and test management in software testing, a wealth of information to be drawn upon in areas where ISO 3450x lacks details. Additionally, the structured use of high-dimensional ODD parameter data for automated testing in automated driving supports the data-driven intelligent transportation systems approach, which leverages diverse, large-scale data to enhance safety, efficiency, and decision-making [26].

Fig.1 schematically illustrates four main stages of a scenario-based safety assurance process focusing on putting the test case allocation method in context, in line with approaches such as [4, 20, 23]. The first stage, Scenario Identification

(Fig.1 Stage 1), defines the ODD and the system's expected behavior. Relevant scenarios are sourced from a database and aligned with test objectives. The second stage, Concrete Test Design (Fig.1 Stage 2), involves translating these high-level scenarios into detailed test cases and specifying the necessary test environment and data requirements, following guidance from ISO 29119 [11]. A test specification encompasses all test design elements, including the test cases, procedures, and requisite environments.
The third stage, the Test Case Allocation Process, addresses the growing need to manage large, parameterized test suites and integrate evidence from multiple test environments. Test environments are generally categorized as field testing, test track testing, XiL testing, or fully virtual testing, each having different attributes.

The allocation aims for effectiveness—ensuring that tests produce credible safety evidence—and efficiency—matching scenarios to environments suited to the required capabilities. Readiness reports (as described in ISO 29119) record environment status, data availability, resource planning, scheduling, risk assessments, and operational constraints. As aims for a method that focuses on safety and needs to be agnostic to the technology up to point in interface with different ODD parameters, in contrast to the similar methods proposed by Striemle et al. [19].
The final stage, Test Execution (Fig.1 Stage 4), proceeds once test cases have been allocated to specific environments. It involves verifying the environment and data are ready, executing test cases, and reporting the results. As exemplified in Section 4, environments should maintain validated parameter ranges, repeating tests that exceed or approach these boundaries in more reliable settings to ensure credible outcomes. Machine-readable scenarios and ODD specifications reduce errors in preparation and execution by confirming that collected data meets the requirements for evaluation and coverage.

3 Methodology for Test case Allocation based on a Formalized ODD (METAFODD).

In the context of the construction of a test case allocation methodology, we leverage an ODD taxonomy construct consistent with ISO 34503 [10], as well as the formalizing ODDs by the use of the Pkl [1] language, as proposed by Skoglund et al. [18]. From that work, we have a hierarchical taxonomy ODD definition, an inclusive ODD, where parameters must be explicitly specified. Our work of refining test attributes into a minimal essential set for the initial allocation process is detailed in [5], emphasizing the key factors required to achieve the intended evaluation objectives. **Test environment attributes**: These include several aspects related to the capacity of the testing system:

– **Safety Hazard Mitigation Capability**: The ability to minimize potential hazards, which could pose risks to participants, including safety drivers and experiment observers, commonly associated with track testing.

- **Test Complexity Capability**: The degree of complexity involved in testing, including the facility's ability to accommodate diverse test elements, orchestration, and ODD conditions.
- **Test Environment Fidelity Capability**: The accuracy with which test models replicate real-world conditions, including vehicle and road user behavior, relevant to the test coverage item, i.e., what you are testing.
- **System Under Test (SUT) Fidelity Capability**: A metric that assesses the abstraction between a model and its intended production implementation, considering the limitations of virtual environments or test harnesses relevant to the test coverage item.

The ODD template is extended with four additional test environment attributes. These attributes must be specified both in the test case definition, which represents the requirements and in the test environment capabilities, which represent the provider. Both sides use the same extended template to ensure comparability for validation. Each of the four test attributes is subdivided into low, medium, and high levels, reflecting incremental capability, where higher levels include the properties of the lower levels. In PKL, this extension can be represented as an addition to the ODD, as illustrated in Fig. 2. Low generally indicates minimal emphasis or significant abstraction, medium corresponds to partial coverage or moderate complexity, and high denotes thorough hazard management or near-complete fidelity.

In a typical virtual environment, safety hazard mitigation and overall throughput are often high because there is no kinetic energy, and multiple tests can run in parallel. However, environment fidelity and SUT fidelity are usually lower owing to abstracted models. In typical XiL setups safety mitigation remains high, throughput is medium, and test complexity is moderate, although environment fidelity typically remains low and SUT fidelity is high. Proving ground tests usually provide a high environment and SUT fidelity because they involve real vehicles and conditions. However, safety hazard mitigation and test throughput remain low, and the practical challenges of physical testing constrain test complexity. Limited safety hazard mitigation capabilities indicate that certain high-risk tests may be infeasible and should not be conducted. This classification scheme is acknowledged as a preliminary. With the prospect of more quantitative metrics [3] there is an opportunity to refine these categories in future research. Nonetheless, even this coarse extension to the ODD has proved beneficial in practice, verifying the soundness of pre-existing (initial) allocations scenario coverage within the ODD.

Good maintainability is achieved as the Pkl templates enable reuse by importation. An example is in Fig.2 where one large module is split into multiple smaller ones, templates can be repeatedly turned into concrete configurations by filling in the blanks and, when necessary, overriding defaults. One can generate static configurations in one of many standard formats to configure testing tools from

```
#ModuleInfo { minPklVersion = "0.25.1" }
module ODD.ODD_template.pkl

import "dyn_template.pkl"
import "env_sun_ext_template.pkl"
import "scen_template.pkl"

open class odd {
  scenery: scen_template.scenery
  environment: env_sun_ext_template.environment
  dynamic: dyn_template.dynamic_elements
}

class ext_odd extends odd {
  #   1 Low, 2 Medium , 3 High
  Safety_Hazard_Mitigation: Int (isBetween(1,3))
  Test_Complexity: Int (isBetween(1,3))
  Test_Environment_Fidelity: Int (isBetween(1,3))
  SUT_Fidelity: Int (isBetween(1,3))
}
```

Fig. 2: Extend the PKL formalized ISO 34503 template with four test environment attributes, specified in both test case requirements and environment capabilities for valid comparison.

this dynamic base directly. The constructed ODD templates in Pkl can be found here [16].

4 Case Study: Reversing Truck Functionality

A case study on automated reversing of a semitrailer truck, further detailed in [6], demonstrates how ODD parameters shape the allocation of test cases. Confined areas with perimeter protections and reduced unauthorized entry risks provide a well-defined operational scope to validate automated functionality in heavy vehicles. Our use case is an automated docking function of a truck to a logistic port, where the area behind the truck is monitored by a camera mounted on the hub. The camera aims to ensure the safety zone (Fig 3) is free from persons and objects. The system is defined to work during the daytime. The daytime test space and the fixed mounting of the camera will then incorporate the special problem of sun glare as defined in Fig 3, which affects object detection.

In many cases, oblique angles just outside or near the field of view are the most prone to inducing reflections or scatter that manifest as glare [14]. Glare can occur over various angles depending on lens design, coatings, and light source intensity, and it must be tested in a high-fidelity environment; in this situation, a simulated environment cannot produce reliable results (see Fig. 4a compared to

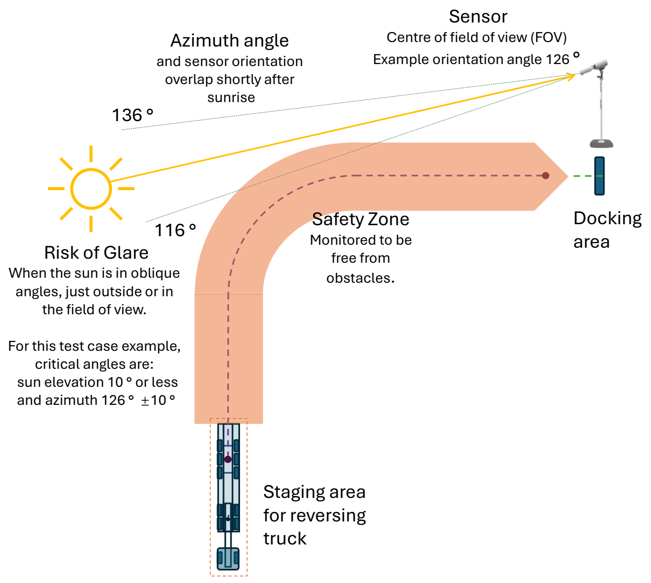
Sensor
Centre of field of view (FOV)
Example orientation angle 126 °
Azimuth angle
and sensor orientation overlap shortly after sunrise
136 °
116 °
Risk of Glare
When the sun is in oblique angles, just outside or in the field of view.
For this test case example, critical angles are:
sun elevation 10 ° or less
and azimuth 126 ° ±10 °
Safety Zone
Monitored to be free from obstacles.
Docking area
Staging area for reversing truck

Fig. 3: Test configuration for a fixed mounted camera.

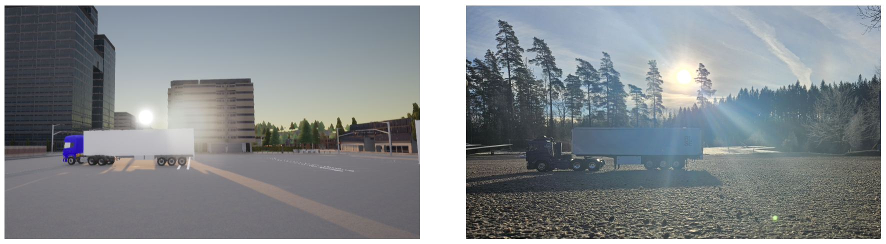

Fig. 4: Illustration of low sun glare test scenarios: the simulation environment shown in (a) uses CARLA with a sun elevation of 6°, while the hardware-in-the-loop scale truck setup in (b) has a sun elevation of 9°.

Fig. 4b), so a hardware-in-the-loop (HiL) environment will be employed. Here, camera orientation, combined with the sun's azimuth and orientation angles, defines a field of view that forms a test subspace. This expansion of the ODD to include the sun position is reflected in the ODD template and, therefore, in the test environment requirements, as shown in Fig. 5.

Fig. 5 uses the template in Fig.2, configured as a test environment requirement, exported to YAML format, and visualized in PlantUML [22], which allows human reviewers to verify the requirements easily.

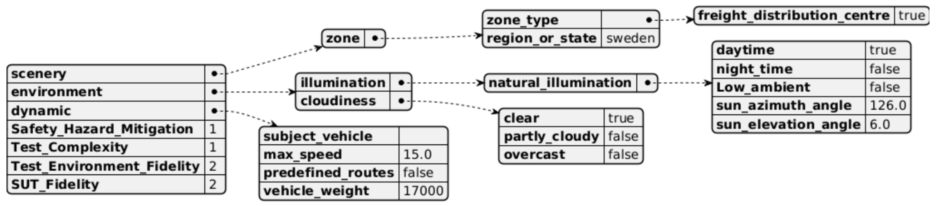

Fig. 5: A subset of the test environment requirement parameters, out of 300 ODD configurable elements.

4.1 Test environment capability, CARLA simulator

The CARLA simulator [21] is an open-source platform used in automated driving research, offering flexibility in modeling a variety of road, weather, and lighting conditions. Fig.6 shows a subset of configurable weather parameters in CARLA, including sun azimuth and elevation angles. The test environment capabilities do include these parameters. Still, correctly modeling the SUT is essential to provide a reliable object detection test result, particularly when evaluating glare effects at oblique angles. Fully replicating complex glare conditions in simulation can be challenging, so a HiL environment will be employed where glare might be an issue to ensure reliable results.

```
#include <WeatherParameters.h>
WeatherParameters (
...
float in_cloudiness
float in_sun_azimuth_angle,
float in_sun_altitude_angle,
#Same as sun_elevation_angle in ODD definition
...)
```

Fig. 6: Excerpt of weather parameters, it is available in CARLA.

To capture the glare caveat for oblique angles, the Fig.7. extends SUT_Fidelity with the conditional expression to incorporate a check on sun_azimuth_angle, ensuring the value lies within 126.0° ± 10.0°. It sets SUT_Fidelity to 1 when both sun_elevation_angle are less than or equal to 10.0°, and sun_azimuth_angle remains in the given range and otherwise sets it to 2. An ODD is a complex class with a deep, non-static structure, allowing for the amendment and extension of its leaves. These leaves can take various forms, including booleans, strings, durations, data sizes, floats, and integers.

Because the ODD structure is large and complex, verifying whether one configured ODD is contained within another—such as comparing specific test require-

```
import   "ODD_template.pkl"

odd_cap_carla: ODD_template.ext_odd = new {
  scenery {
    zone {
      region_or_state = "sweden"
      zone_type {
        freight_distribution_centre = true
      }
    }
  }
  environment {
    illumination {
      natural_illumination {
        #  Max capability
        sun_azimuth_angle = 360.0
        sun_elevation_angle = 90.0
      }
    }
  }
  Safety_Hazard_Mitigation = 3
  Test_Complexity = 3
  Test_Environment_Fidelity = 2
  #  Glare caveat for oblique angles
  #  When the risk of glare SUT_Fidelity = low
  SUT_Fidelity = (if (
    (odd_req.environment.illumination.natural_illumination
    .sun_azimuth_angle >= 116.0)
    && (odd_req.environment.illumination.natural_illumination
    .sun_azimuth_angle <= 136.0)
    && (odd_req.environment.illumination.natural_illumination
    .sun_elevation_angle <= 10.0)
) 1 else 2)
}
```

Fig. 7: The CARLA test environment capability.

```
# The ext_ODD_test contains test requirements and test capabilities.
# Also a method of generic comparisons that evaluate those conditions.
...
Within_CARLAs_Capabilities = genCompare.apply(odd_cap_carla, odd_req)
Within_Scaletruck_Capabilities = genCompare.apply(odd_cap_scale, odd_req)
...
C:\pkl\ODD_allocate> ./pkl eval .\ext_ODD_test.pkl
...
#  Result
Within_CARLAs_Capabilites = false
Within_Scaletruck_Capabilites = true
```

Fig. 8: Automatic allocation evaluation.

ments in Fig. 5 with the CARLA test environment capabilities in 7—necessitates tool support. A validation method called *genericCompare* is defined using the reflection property of Pkl 8 [16]. Reflection enables querying a program's metadata, such as the classes within an assembly and the methods, fields, and properties they contain. By leveraging this capability, an intelligent recursive loop can be constructed to perform a detailed, piecewise comparison of all leaves. This method ensures that string and boolean values are checked for equality while integers and floats are compared using an equality or "less than" condition.

The proposed method for comparing two configured ODDs has several limitations. One significant limitation is handling extremes such as temperature at both ends of numeric ranges, which needs to be addressed. Simply checking for equality or "less than" conditions may not capture the nuances of overlapping ranges or boundary conditions, limitations inherited from the specification, and also best addressed at that level. Limitations aside, the method works and can be used for both automation and assessment of allocations.

Test criteria outlined in Section 3 can be integrated with the template in Fig.2. and, in conjunction with the genericCompre function (Fig.8.), enable the comparison of test requirements (Fig. 5) with environment capabilities, such as those in Fig.7. These elements, when integrated, form a prototype methodology for automatically allocating test cases to suitable environments.

5 Conclusions

In conclusion, any ODD definition formalized using the Pkl language method [18] can be extended with test environment attributes to capture test environment capabilities better. This enables automated, flexible, and scalable test allocation.

This framework-agnostic approach aligns with multi-pillar validation strategies such as NATM [4] and SUNRISE [13], making it compatible with assurance cases that rely on heterogeneous evidence from diverse test environments. The representation permits verification of whether one ODD is subsumed by another, demonstrating its scalability and efficiency in handling extensive ODDs and ability to handle scenarios requiring finer-grained environment attributes.

We propose and provide [16] an approach that extends the ODD [10] formalization in the Pkl configuration language by incorporating test environment attributes and tools for automated test case allocation, facilitating systematic and data-driven matching of scenario requirements to environment capabilities. Although still a proof of concept, this approach establishes a foundation for further refinement and broader adoption through community collaboration. Its continued development may benefit developers, assessors, tool vendors, and standardization bodies, and has the potential for wider use if its value is recognized by the research community.

Future work will examine domain-specific ODD definitions—such as those in forestry—and expand the formalization to generate test spaces that facilitate automated allocation. Efforts will also include investigating compatibility with OpenODD [2] to ensure alignment with emerging ASAM standards and explore potential integration opportunities.

Acknowledgments. We acknowledge the support of the Swedish Knowledge Foundation via the industrial doctoral school RELIANT, grant nr: 20220130. This research was carried out within the SUNRISE project and is funded by the European Union's Horizon Europe Research and Innovation Actions under grant agreement No.101069573. However, views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Union's Horizon Europe Research and Innovation Actions.

Disclosure of Interests. The authors have no competing interests to declare relevant to this article's content.

References

1. Apple Inc.: Pkl :: Pkl Docs. https://pkl-lang.org/ (2025)
2. ASAM: ASAM OpenODD: Concept Paper (2021)
3. Böde, E., Büker, M., Eberle, U., Fränzle, M., Gerwinn, S., Neurohr, B.: Efficient splitting of test and simulation cases for the verification of highly automated driving functions:. In: 37th International Conference, SAFECOMP 2018, Västerås, Sweden, September 19-21, 2018, Proceedings. pp. 139–153 (Sep 2018). https://doi.org/10.1007/978-3-319-99130-6_10
4. ECE/TRANS/WP.29/2021/61: (GRVA) New Assessment/Test Method for Automated Driving (NATM) - Master Document | UNECE (2021)
5. Hillbrand et. al.: D3.3 Report on the Initial Allocation of Scenarios to Test Instances | Sunrise Project (Feb 2025)
6. Hillbrand et. al.: D7.2 Safety assurance framework demonstration instances design | Sunrise Project (Feb 2025)
7. ISO: ISO 26262:2018 Road vehicles – Functional safety (2018)
8. ISO: ISO 34501 Road vehicles — Road vehicles — Test scenarios for automated driving systems — Vocabulary (2022)

9. ISO: ISO 34502 Road vehicles — Test scenarios for automated driving systems — Scenario based safety evaluation framework (2022)
10. ISO: ISO 34503 Road Vehicles — Test scenarios for automated driving systems — Specification for operational design domain (2023)
11. ISO/ICE/IEEE: ISO/ICE/IEEE 29119-1:2022 Software and systems engineering - Software testing (2022)
12. Kalra, N., Paddock, S.M.: Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? Transportation Research Part A: Policy and Practice **94**, 182–193 (Dec 2016). https://doi.org/10.1016/j.tra.2016.09.010
13. Project, S.: Sunrise Project | Developing and providing a harmonized and scalable CCAM Safety Assurance Framework (2025)
14. Ray, S.F.: Applied Photographic Optics : Lenses and Optical Systems for Photography, Film, Video, Electronic and Digital Imaging. Oxford : Focal (2002)
15. Riedmaier, S., Ponn, T., Ludwig, D., Schick, B., Diermeyer, F.: Survey on Scenario-Based Safety Assessment of Automated Vehicles. IEEE access : practical innovations, open solutions **8**, 87456–87477 (2020). https://doi.org/10.1109/ACCESS.2020.2993730
16. Skoglund, M.: Baseline test case allocation using an ODD. https://github.com/Marskse/ODD_ext
17. Skoglund, M., Warg, F., Thorsén, A., Bergman, M.: Enhancing Safety Assessment of Automated Driving Systems with Key Enabling Technology Assessment Templates. Vehicles **5**(4), 1818–1843 (Dec 2023). https://doi.org/10.3390/vehicles5040098
18. Skoglund, M., Warg, F., Thorsén, A., Hansson, H., Punnekkat, S.: Formalizing Operational Design Domains with the Pkl Language (2025)
19. Steimle, M., Weber, N., Maurer, M.: 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 : practical innovations, open solutions **10**, 6260–6285 (2022)
20. SUNRISE project: SUNRISE Safety Assurance Framework - High-Level Overview. https://ccam-sunrise-project.eu/high-level-overview/
21. Team, CARLA.: CARLA. http://carla.org//
22. The plantuml project: Open-source tool that uses simple textual descriptions to draw beautiful UML diagrams. https://plantuml.com/
23. Thorn, E., Kimmel, S.C., Chaka, M., Virginia Tech Transportation Institute, Southwest Research Institute, Booz Allen Hamilton, Inc.: A Framework for Automated Driving System Testable Cases and Scenarios. Tech. Rep. DOT HS 812 623, NHTSA (Sep 2018)
24. UNECE: UN Regulation No 157 – Uniform provisions concerning the approval of vehicles with regards to Automated Lane Keeping Systems [2021/389] (Mar 2021)
25. Weissensteiner, P., Stettinger, G., Khastgir, S., Watzenig, D.: Operational Design Domain-Driven Coverage for the Safety Argumentation of Automated Vehicles. IEEE access : practical innovations, open solutions **11**, 12263–12284 (2023). https://doi.org/10.1109/ACCESS.2023.3242127
26. Zhang, J., Wang, F.Y., Wang, K., Lin, W.H., Xu, X., Chen, C.: Data-driven intelligent transportation systems: A survey. IEEE Transactions on Intelligent Transportation Systems **12**(4), 1624–1639 (2011). https://doi.org/10.1109/TITS.2011.2158001