ASAM

Association for Standardization of
Automation and Measuring Systems

EVOLVING LANDSCAPES OF COLLABORATIVE TESTING FOR ADAS & AD

ASAM Test Specification Study Group Report 2022

Version 1.0.0

Date: 2022-02-02

Table of Contents

Foreword

A main focus of collaborative efforts across the automotive industry needs to be testing strategies and how to adapt them.

Software-defined vehicles and autonomous driving functions are continuously and drastically increasing the complexity of testing. To tackle this challenge, a network of experts across all branches of the automotive industry have joined forces under the umbrella of ASAM e.V.

The ASAM Test Specification Study Group created a comprehensive overview of the testing and standardization landscape and developed a test methodology blueprint that embraces the multi-pillar approach needed to prove the safety of autonomous driving functions. A potential **basis for future standardization**. The results are a statement, a recommendation, and a definite invitation to shape this future together.

1 Executive Summary

Although sometimes it may feel like we have a frustratingly limited number of ways to steer a vehicle through traffic, the reality is that driving is tremendously complex, with an infinite number of possible scenarios – particularly when it comes to testing requirements. One question the current developments will soon pose is: who exactly is behind the steering wheel? Today, we see the adoption of SAE Level 0 to Level 2 advanced driver assistance systems (ADAS) throughout the industry, including features such as Brake Assist, Lane Keeping Assist, Electronic Stability Control, Blind Spot Indication, or Parking Assist. While these features do not eliminate the role of the driver, the interaction with the environment increasingly transfers from driver to vehicle. In other words, the transition from assisted driving to fully automated driving is on the horizon. Consequently, the automotive industry is undergoing the biggest changes in its history. This paradigm shift affects everyone and brings opportunities and challenges to the industry.

Levels of Autonomy and the Difficulty of Proving Safety

The future of transportation promises to make life safer and more mobile for everyone, with positive economic results. But to realize that promise, it is first necessary to test new vehicles and every part of their architecture, especially as the parts become smarter and highly complex. However, the increased complexity necessitates a radical change of test methods and new concepts for comprehensive vehicle validation in both the physical and the virtual world. Still, many players appear to have neglected to consider the best path for developing and implementing suitable strategies. Without toolkits independent of manufacturers or domains, and without intensive exchange across all subsectors, highly autonomous driving simply cannot be realized. We are on the verge of ushering in a new era of transportation safety through autonomous mobility, but this shift raises big questions. What constitutes state-of-the-art testing of driving functions in a rapidly evolving environment? How will the automotive industry a) consider best practices for testing and safe deployment while b) at the same time integrating new technologies like Artificial Intelligence and c) complying with regulations that are not yet developed? In February 2021, the UNECE (United Nations Economic Commission for Europe) presented the New Assessment/Test Method for Automated Driving (NATM) – a framework and potential game changer. Its multi-pillar approach considers the topic's high level of complexity and poses new opportunities for more comprehensive requirement sets as a focus of technical services. However, these still need to be created and standardized.

Why the Joint Effort?

To expedite this process, a transdisciplinary network of experts was formed, the ASAM Test Specification Study Group, including representatives of technical services, software providers, manufacturers and OEMS, engineering and test data management professionals.
Under the banner of ASAM e.V. experts have collaborated for more than twenty years in order to create standards for the development and testing of automotive electrical/electronic systems. They succeeded in reducing costs and efforts at OEMs and Tier-1 companies for the development, maintenance, and administration of development tool chains. Currently the ASAM portfolio

comprises more than 30 trusted standards that are applied in automotive development across the globe. While this means that many pieces of the puzzle when it comes to standardized scenario-based testing workflows are being tackled, the overall interplay between those pieces had not been examined in such close detail. The following report documents the results of this new joint effort and aims to define a valid basis for follow-up activities and projects.

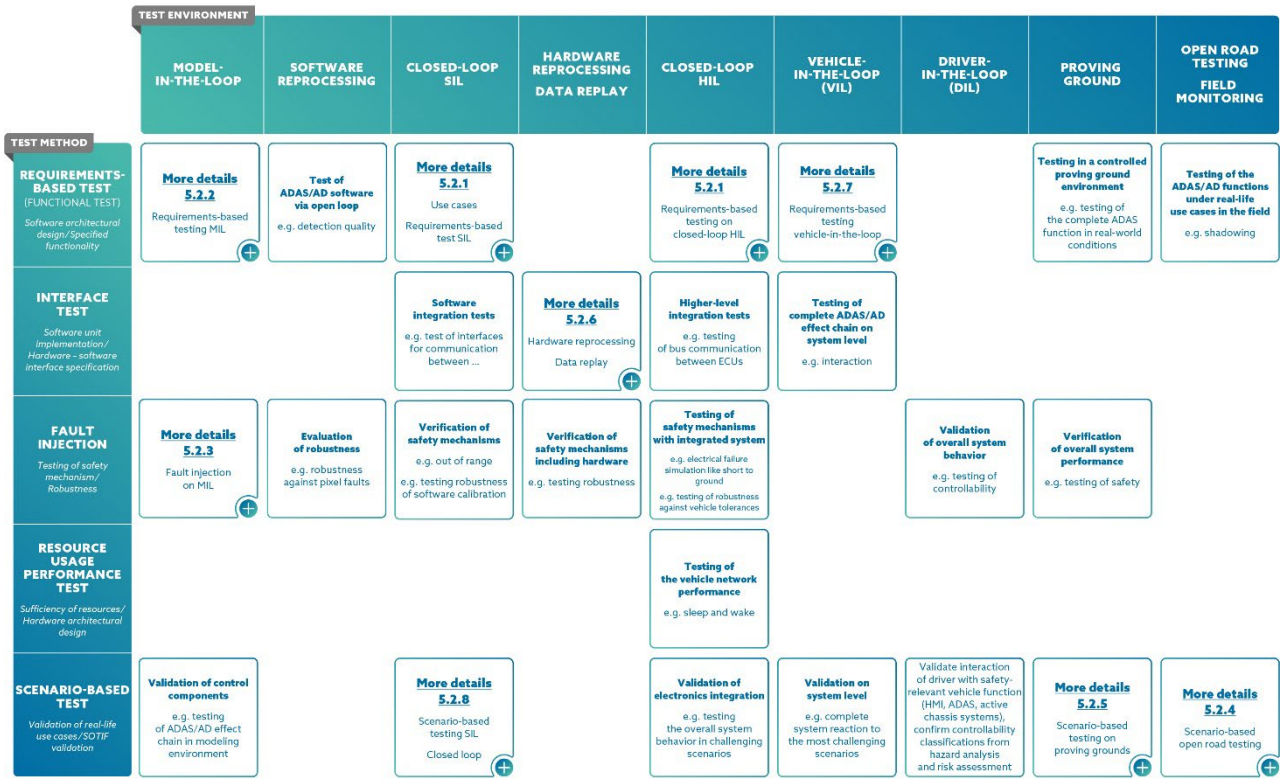
Main goals of the ASAM Test Specification Study Group project:

- Provide overview of test methods in the field of ADAS/AD
- Develop a potential basis for future testing, the Test Strategy Blueprint
- Detailed use cases for the implementation of a test strategy
- Alignment with current standardizations
- Provide recommendations for stakeholders and proposals for further standardization activities

A New Strategy Blueprint

Based on the experience of the many experts working together during the course of the ASAM Test Specification Study Group project, a blueprint to meet the challenges of testing has been developed. Test strategies are becoming more complex, but this blueprint makes it possible to set a sensible starting point. A holistic best practice that can be tailored according to the specific requirements of manufacturing and other projects, but one that meets regulatory, legal, and technical requirements. This blueprint is to be understood as an invitation. An invitation to use, develop, and critically engage with. An opportunity to work together on the challenges of an increasingly complex automotive industry.

The implementation of new testing strategies is an intricate task. Therefore, we defined concrete workflows and identified the need for a uniform test data management. In addition, we provide initial solutions so that users are not left alone. Despite not being able to analyze and highlight all aspects, we were able to define concrete recommendations for action and follow-up activities. Current test strategies, which are often heterogeneous and have grown over several vehicle generations, must be viewed holistically and also used during vehicle approval. The various test procedures and test environments highlighted in the blueprint can also be clearly anchored in the data-driven process. It is important to understand that the phases are no longer strictly separated from one another, but that transitions are smooth and continuous. Iterations and changes between the phases occur constantly and merge into one another. In the following chart you can see a possible and reasonable combination to fulfill test coverage for a software-centric and (partially) autonomous vehicle, which is sufficient for release and homologation.



The derived blueprint, exemplifying test coverage for a software-centric vehicle

The Vision of Seamless Exchange

The following chapters and sub-sections elaborate on the current standardization landscape and possible future developments, describe relevant use cases and how the new test strategy blueprint can be used. The report aims to provide overview and to additionally clarify the role of data-driven development and the importance of test data management. The final chapter summarizes the study group's findings, deducing recommendations for further action. Competition stimulates progress. If a specific technology is not effective for its users, however, competition loses its benefits. It leads to incompatible systems rather than better products. That is one reason why ASAM pursues a vision of the free interconnection of development process chain tools and the seamless exchange of data. The combined efforts of the ASM Test Specification Study Group bear witness to the fertility of this concept.

2 Scope

Mapping a Changing Landscape

When attempting to map the standardization landscape in its entirety, the first step is to define what is meant by “entirety.” Identifying the boundaries is an important part of the process, before examining the relevance of all aspects to achieve a comprehensive overview. The study group decided on a sequential project approach, a phase and milestone plan with intermediate results that build on each other. Methods and use cases were clustered to facilitate structure and exchange in sub–workgroups.

The report examines the relevant test techniques and use cases for testing and homologation of the ADAS/AD Domain in the automotive industry with a special focus on software-centric cars with modern E/E architectures. The goal is to identify relevant standards, potential workflows, and their variants, and the overall interplay between these parts to form a cohesive whole.
This analysis leads to a documented set of overarching use cases for testing and homologation, a set of potential workflows implementing them, and an overview of relevant users, standards, and their application.

Additionally, the study group identifies gaps in the workflows, leading to the identification of potentially necessary additions to existing standards, liaisons between standards, or completely new standards.

Recommendations for these standards or additions are collected and documented. The goal is to define a valid basis for follow-up activities and projects. As the safety arguments and homologation of modern vehicles are the key aspects, security-specific aspects such as fuzzing, pen testing, and TARA-driven testing are not currently receiving much attention.

In order to cover a large part of the automotive test landscape in the ADAS/AD domain, the group pursued a broad scope when considering possible test techniques and test environments. The goal is to prioritize these test techniques and to map them to user journeys and workflows in order to map their relevance (and also for standardization). It is important to us that we cover large parts of the test map, even though we will focus on certain aspects as we go along. For follow-up projects, we believe it is necessary to provide as complete an overview as possible. From our point of view, the test procedures presented represent a large part of the relevant test procedures, which, through combinations of test environments, enable us to provide an almost complete representation of the relevant tests from testing during development to homologation.

3 Terms and Definitions

An Outline of Crucial Concepts

Based on the following definitions, the study group were able to derive a structure and find an adequate level of abstraction to measure up to the complexity of the topic as well as to the applicable compilation of a set of methods, use cases and test aspects.

3.1 Test Method

The term “test method” is not defined in any of the common standards like the ISTQB glossary or ISO 29119, but there are some hints how the term could be used. The American Society for Testing and Materials defines “test method” as :

> ***“… a method for a test in science or engineering, such as a physical test, chemical test, or statistical test. It is a definitive procedure that produces a test result. In order to ensure accurate and relevant test results, a test method should be ‘explicit, unambiguous, and experimentally feasible,’ as well as effective and reproducible.”***

Similarities to other standards like software methodology are object-oriented, sequential, data-driven, and agile software development methods. In the context of highly automated driving we speak also about the PEGASUS Method for Assessment of Highly Automated Driving Function (HAD-F) propagated from the PEGASUS project (https://www.pegasusprojekt.de/en/).

A test method is any procedure that fulfils test goals and defines, for example, applicable test techniques or practices as a part of this specific method. Following this idea, requirements-based testing, for example, is a test practice, but not a test method in itself. There are many methods to conduct requirements-based tests to obtain test results and fulfil test goals.

A test method may consist of

- Choice of test practices (requirements-based testing, scenario-based testing, model-based testing)
- Test level (system test level, integration test level)
- Test environments (HIL, SIL, MIL, VIL, proving ground, real road testing)
- Test techniques (equivalence partitioning, state-based testing, failure injection test (?)
- Applicable procedures/processes (postprocessing, risk assessment)
- Choice of coverage criteria

So, a test method represents a partial implementation of the test strategy for an aspect of the test item to be tested.

3.2 Test Case

Test cases are one of the most important building blocks of automotive testing and are used in all different kinds of testing methodologies. A test case defines an autonomous unit that describes one particular test that can be specified, executed, and assessed.

In the following, we will present three different but well-used definitions of “test case” and compare them. We will not choose the right definition out of these terms, since they are quite similar and the exact choice depends on usage.

Using the current definition of the International Software Testing Qualification Board (ISTQB), a test case can be defined as:

> ***“A set of preconditions, inputs, actions (where applicable), expected results, and postconditions, developed based on test conditions.”***

The current ISO standard (ISO 29119) omits the postconditions (which are optional and could be results of the test) and defines a test case as:

> ***“A set of test case preconditions, inputs (including actions, where applicable), and expected results, developed to drive the execution of a test item to meet test objectives, including correct implementation, error identification, checking quality, and other valued information.”***

This can be shortened to:

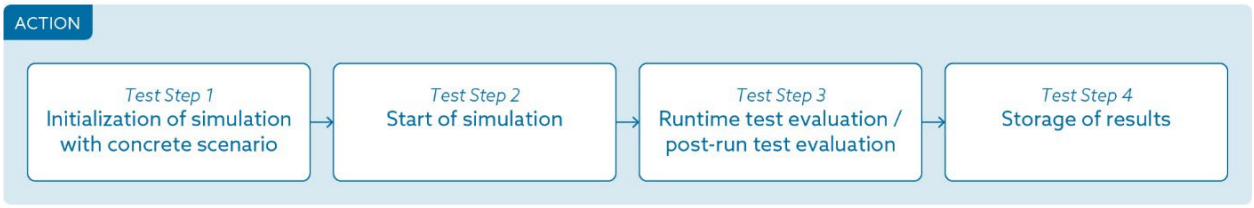
> ***“A set of preconditions, inputs, and expected results, developed to drive the execution of a test item to meet test objectives.”***

The previously mentioned actions are part of the general inputs of the test case. The latter definition is a very probable candidate for being used in the new ISO standard.

As stated in the second paragraph, these three definitions are quite similar, with only the necessity of stating postconditions and actions as part of a test case being debatable and without effect on the usage of the term in the rest of this document.

If a scenario-based test approach is used test cases may reference scenarios for environment simulation.

Example of a Scenario-Based Test



Referenced Standards:

- ISTQB
- ISO/IEC/IEEE 29119

3.3 Scenario

A scenario describes how the world or scene changes with time, usually from a specific perspective. In the context of vehicles and driving, this will encompass the developing view of both world-fixed elements such as the road layout and static furniture, world-changing elements such as weather and lighting, and dynamically moving elements such as other vehicles, objects, people, or animals. In addition, a scenario may contain scenario validity criteria.

At the time of writing, the group were made aware of a new definition for a test scenario defined in ISO 34501. This may have an impact on future revisions of the workflows defined herein, but it was not considered for this revision.

3.4 Test Specification

Test case specifications are important documentation types in automotive testing across all different kinds of testing methodologies (from requirements-based to interface testing, fault injection, resource usage, and so on) and have a direct impact on the testing process.

The test specification describes everything that is required for the test execution. This includes both the test environment and the sequence that describes the test execution. As part of the testing process, several steps are carried out and include the following activities:

- Test planning
- Test monitoring and control
- Test analysis and test design
- Test implementation
- Test execution
- Test completion

So, the test specification is the complete documentation of test design and test implementation and includes test cases and other necessary information to run those test cases. As part of the start-up phase, the preparations, follow-up work, and actions for "cleaning up" are defined. In the context of test design, the focus is on identifying and defining the required test cases. The aim is to define what exactly we want to test and to recognize defined scenarios. As a result, we get a wrapper for the test cases as the sequence to be executed.

For this purpose the generic structure of the test specification is broken down into three components.

1. The first of them is the test model specification or test design specification, which describe the target of testing and the model used by the tester to derive test cases:
    a. Traditional feature sets
    b. Test coverage items
    c. Test models
2. The test case specification, which describes the test cases
3. The test procedure specification, which describes how test cases are grouped with additional information necessary to perform testing

The ISTQB standard defines the test specification as a document that consists of a test design specification, test case specification (describing one or more test cases), and/or test procedure specification.

Referenced standards:

- ISTQB
- ISO/IEC/IEEE 29119
- Retired IEEE 829

3.5 The Relation between Test Case and Scenario

Test case specifications and scenario descriptions are important document types in automotive testing across all different kinds of testing methodologies (from requirements based to interfaces testing, fault injection, resource usage, and so on).

For example, in requirements-based testing methodology, the use of test case descriptions has proven successful in specifying requirements, in test bench configuration, and in defining test steps and expected result of test runs. As another example, scenarios have become increasingly important in the context of scenario-based testing using simulation (also known as virtual testing) and other relevant test environments for the homologation of ADAS/AD (*reference to the test matrix in the report*). The scenario-based testing methodology for ADS is evolving, and different standards and regulations are being formed with respect to it. Virtual testing environments based on scenario specifications have been demonstrated by few AV-focused companies.

In scenario-based testing, one or more scenarios are referenced from a test case specification. Scenarios contain the behavioral specifications for vehicles, pedestrians, etc. acting in the (virtual) world. Scenarios may include scenario validity criteria that may impact the test results. At the end, there is a wide spectrum of potential mutual relationships between the test and the scenarios. Below are several examples of such relationships:

- The test may affect scenario parametrization by posing additional constraints on it
- The scenario may result in events that are used by the test

And many others.

- There is always a trade-off between the amount of information flowing back and forth between the test and the scenario and the reusability of different scenarios across different tests. That is, independent of the choice of scenario language and test language, the ability to exchange information and reuse tests and scenarios should be allowed as much as possible. This leads to a need for a standardized interface between the test and scenario that clearly specifies the interaction rules.

Standardized Interface

Due to the heterogeneous architecture of both test instances and simulation environments in the industrial environment, both document types must be formulated generically with respect to the underlying hardware and software architecture. This is necessary to enable the consistent use of test cases and scenarios across different test instances and test environments throughout the development process and within homologation processes.

Illustration Examples and Starting Point for Further Discussions

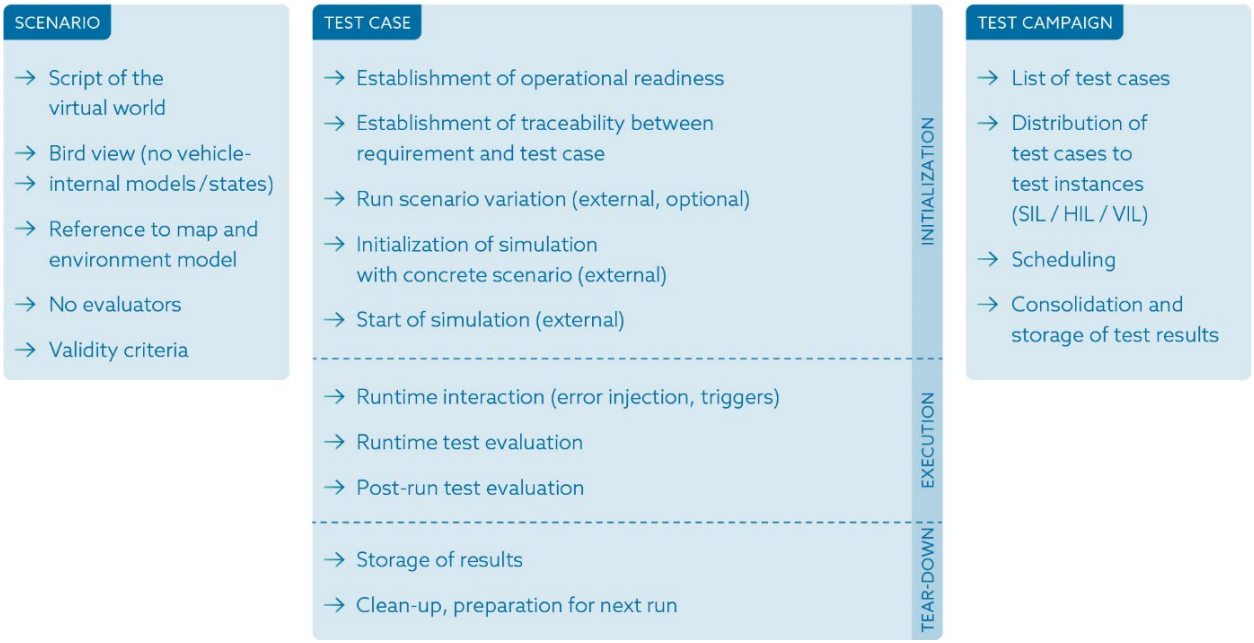
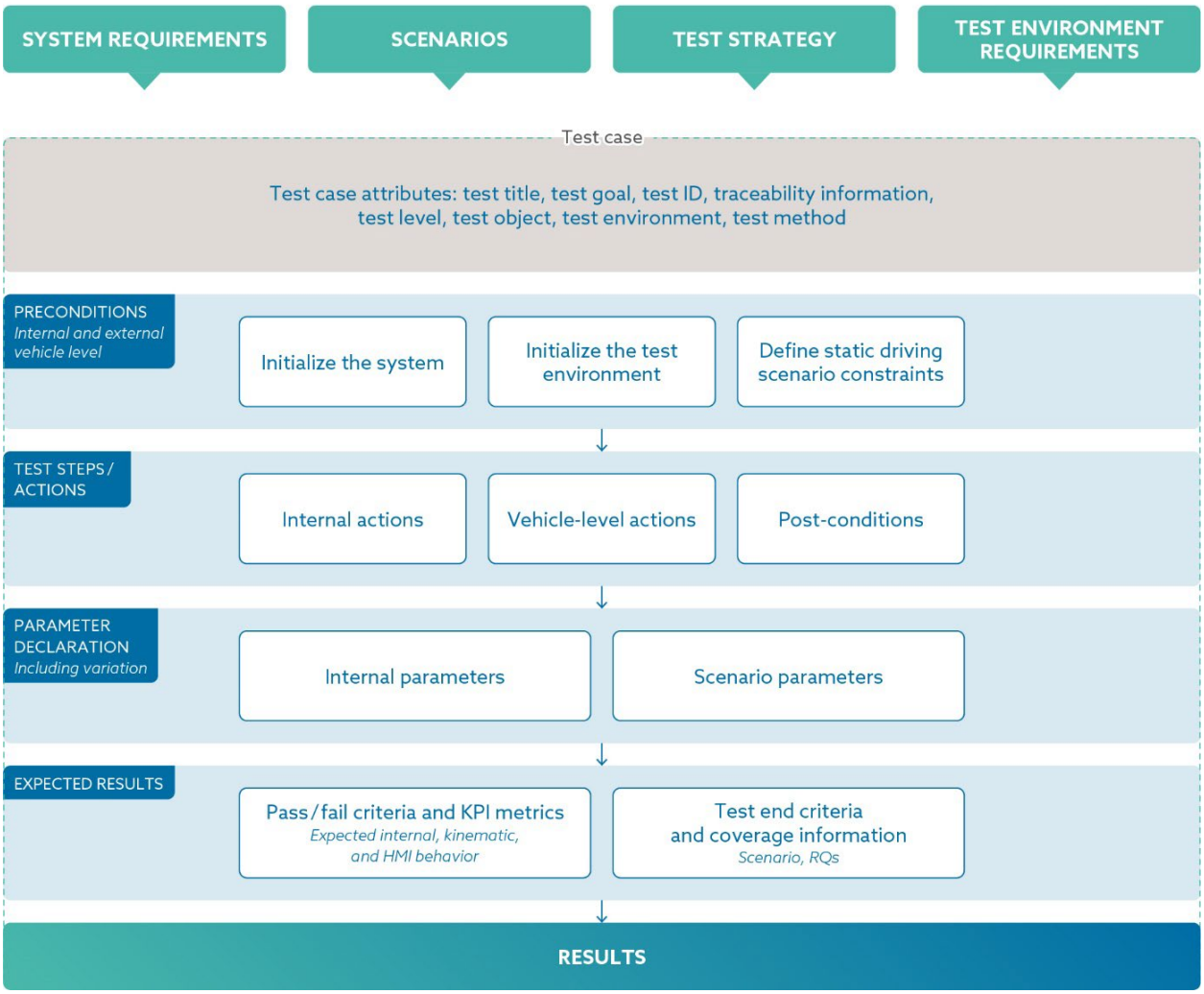


Illustration Examples and Starting Point for Further Discussions



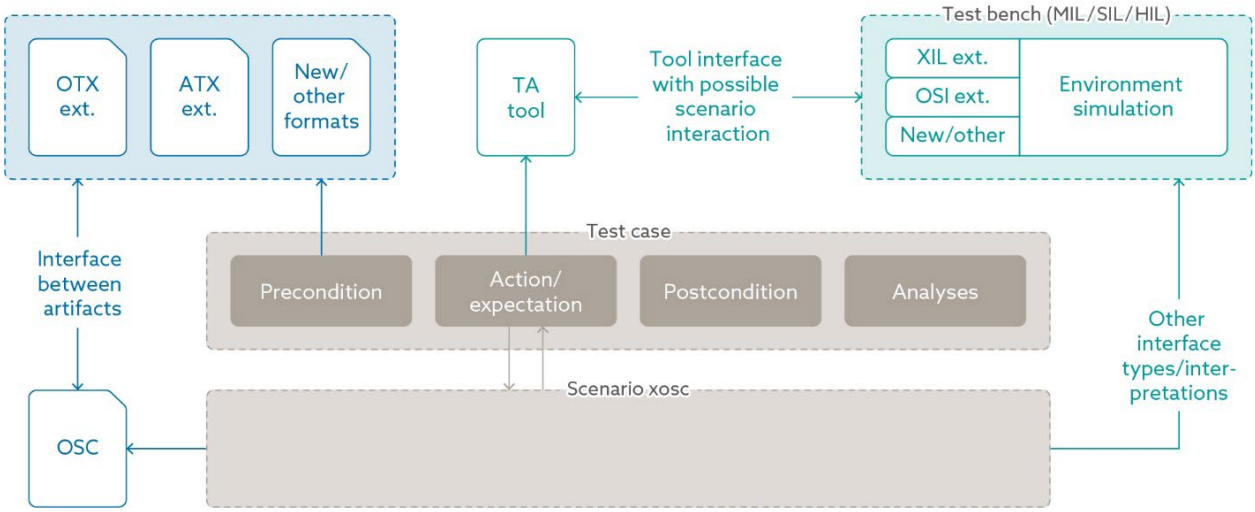
In scenario-based testing, one or more scenarios are referenced from a test case specification. This requires an interface within the test case description to implement higher-level concepts (e.g. parametrization) consistently in tests and scenarios. To enable the use of different scenario description languages (from the ASAM Open Standards), this interface must be designed in an abstract and modular manner. Standardizing the interface simplifies the synchronization of the developments and reduces integration risks.

Minimum requirements need to be defined regarding what should be handled between test case and scenario.

- Reference to scenarios (from within test cases)
- Parametrization and metrics of scenarios and test cases (e.g. changing the initial velocity of a vehicle) shall be consistent

- Specification of interfaces for runtime interaction within test cases
- with scenarios (access to language elements, e.g. events and success criteria)
- with environment simulation and the test environment (interface examples: see image [tracetronic]) e.g. evaluation and modification of numeric values during test execution (model parameters, ECU values, bus data, etc.)
- Interfaces should be bidirectional

Relevant Interfaces in Scenario-based Testing



4 Automotive Industry Insights

The Status Quo of Test Strategies and Homologation Developments

Due to the high complexity of the driving task in a so-called open world, requirement- and specification-driven approaches tend to have limits. Up until now, the set of methods in requirement-based engineering were not able to keep up with this increasing complexity. To address this issue, the term “safety of the intended function” (SOTIF) has been created. In order to recover the reduced capabilities of the left side of the V-model, the activities in the verification and validation branch need to relate to each other in order to provide a solid foundation for the safety argumentation.

4.1 Homologation and the Transforming Role of Technical Services

Out of the lab onto the road?

The New Assessment/Test Method for Automated Driving (NATM) published in February 2021 provides a framework composed of a scenarios catalog and five validation methodologies to which this report refers, underlining the fact that the homologation process will develop further over time. While future strategies and type approval for new vehicles and ADAS functions will vary, the common denominator is certain: today and tomorrow the assessment of an ADS and its ability to demonstrate safe behavior when operating in the real world needs to combine three pillars; real-world testing, physical certification tests, and extensive auditing.

The high complexity of the driving task and the quest for the best approach require a technical inspection to go beyond a static set of test cases. Additionally, the complexity of the automation systems in terms of architecture and components require checks beyond a static set of technical checklist items. To complement well-established static test cases, checks of procedures and fulfillment of objectives are needed (e.g. through audits).

4.1.1 Testing and type approval – current and future regulations

The aforementioned need to relate individual test results to the remaining tests and results finds its expression in the “multi-pillar” approach put forward by GRVA (UNECE Working Party on Automated/Autonomous and Connected Vehicles) and the effort of the New Assessment/Test Method for Automated Driving (NATM) at the UNECE.

Thus, in the latest and in upcoming regulations, one can expect a more comprehensive set of requirements as compared to the past. At this point in time, the following items are discussed:

- Scenario catalog
- Simulation/virtual testing
- Track testing
- Real-world testing
- Audit/assessment
- In-service monitoring and reporting

Note that there are topics that can only be addressed by combinations of the above items. Some examples are:

- Trustworthiness (simulation/virtual testing + track testing + real-world testing)
- Qualification of tools (audit/assessment + testing + in-service monitoring)
- Coverage of hazardous scenarios (audit/assessment + scenario catalog)

- Sufficient exploration of unknown hazardous scenarios (real-world testing + in-service monitoring + scenario catalog)

It is these objectives that are in the focus of inspections by technical service providers.

4.1.2 Possible impact of the study group and defined test strategy blueprint

1. The blueprint of the strategy can be used to argue residual risk estimations, performed scenarios, and considered weaknesses and mitigations
   The aforementioned objective to achieve coverage of all hazardous scenarios is obviously rooted in the need to estimate the residual risk and to evaluate whether the achieved values can be accepted. A test strategy blueprint that links hazard analyzes and exploration of scenarios with the verification of analysis results and effectivity of implemented risk mitigation measures is one key element to achieving the goal of an evaluation of residual risks.

2. A testing strategy blueprint defined by the study group can serve as basis for an assessment of OEM-/Tier-1-specific test strategies
   As outlined above, the verification and validation challenge for automated driving is also rooted in the fact that different tests, which make use of different methods and Test Environments, are interdependent: the full strength of the verification and validation evidence and argument can only be unfolded if the various results make reference to each other. A testing strategy blueprint might show the most relevant types of interrelation. It might thus serve as a basis for setting up a verification and validation strategy at OEM or Tier-1, as well as a basis for assessments and release.

3. The different methods in the test strategy can be reviewed in different ways: document review, witness testing, or retest at technical service
   The assessment of the test activities and test results produced in terms of agreement with the verification and validation strategy, as well as requirements from standards and regulations, will probably require several approaches. These approaches (very much similar to the test methods themselves) have to trade off effort and depth of investigation. On the low-effort end, document reviews might be used to confirm that, for example, passed test results are truly confirming safety and failed tests have been properly fed back into the development cycle. For deeper insight into the tests performed, witnessed tests are especially valuable in case very specific and specialized test equipment is required. To achieve fully independent confirmation of test results on the high-effort end, a fully independent retest performed by a technical service might complete the range of confirmation of the test activities.

4.1.3 Examples with special relevance for type approval

Scenario-Based Approval of ALKS (Automated Lane Keeping Systems) using X-in-the-Loop

Scenario-based test approaches have several features that make them interesting for assessment, release, and type approval of automated driving functions. The first aspect to mention here is the analogies to methods used for risk evaluation. These analogies allow the use of in-the-loop methods to confirm cases identified during analysis. This might include aspects like severity ratings or input parameters for controllability considerations, as well as identification of edge cases.

Secondly, scenario-based in-the-loop methods might be used to build a positive risk argument along the lines of UNECE R157 (ALKS) Annex 4, Appendix 3, by comparing the system of interest with an (existing) benchmark. Such a benchmark might also be a driver model as suggested in UNECE R157.

Thirdly, to confirm the safety of the intended function (SOTIF from ISO/DIS 21448 or “operational safety” in UNECE sources), it will be necessary to argue the absence of unknown hazardous scenarios (refer to “area 3” from ISO/DIS 21448). This endeavor is sometimes also referred to as the exploration of the unknown. In case this exploration requires a parameter variation, in-the-loop techniques should complement real-world observations.

SW/HW Reprocessing/Data-Replay for Perception Systems

With the currently available methods and experiences, the argument to control risk cannot end after product release. Instead, the final evidence for acceptable risk (e.g., in terms of a positive risk balance) can only be provided in the field. Thus, field observation has become an essential part of the UNECE regulations. Due to limitations in connectivity and in-vehicle data storage capacity, testing capabilities to reproduce adverse interactions in real-world application are required to operate an automated vehicle fleet safely. Reprocessing of recorded (or reconstructed) sensor data might allow a reconstruction, analysis, and future mitigation of such adverse effects.

VIL and Proving Ground

Previously, the outstanding role of in-the-loop and reprocessing techniques has been discussed. However, the key element that makes these methods efficient – namely partial modeling (and thus simplification) of the real world – is also a point of weakness once it comes to systematic deviations. To keep these potential systematic issues under control, a close alignment with other test results that are closer to the real-world application of automated vehicles on public roads is required. This means vehicle-in-the-loop and proving ground tests are key to model validation in other in-the-loop approaches. Beyond this, proving ground tests are integrated in the UNECE multi-pillar approach as a mandatory part, requiring the accomplishment of a minimum set of challenges.

4.1.4 The trustworthiness of test environments

ISO 26262 Tool Qualification

As an automation system has control over the major actuations of the vehicle – longitudinal and lateral control – its decisions are safety critical. Thus, automation systems are subject to ISO 26262. As ISO 26262 affects requirements for testing and the use of SW tools that are applied to cover objectives from the standard. Therefore, all parts of the test tool chain used to verify or validate safety-critical decisions will be subject to tool qualification. This means the tool use cases need to be specified and compliance with these specifications has to be shown in agreement with the provisions in ISO 26262.

Model Validation

Similar to the in-vehicle automation function that has functional safety and SOTIF aspects, models used for verification and validations also need to be executed with integrity and need to provide a certain minimum performance in terms of agreement with the real world (e.g. ISO 11010). In the previous paragraph on proving ground tests, it was already outlined that there is the need to support the lower-level in-the-loop tests in terms of validation of the deployed models. By nature, a model is a simplification of the real world. The trick with modeling is thus to remove complexity that is not relevant for the process of interest, keeping the relevant parts sufficiently realistic. It needs to be confirmed whether this abstraction process has been successful. This confirmation is typically done by validating the models. This means the model results are confronted with results obtained in real-world experiments. Only with such input on the trustworthiness of the tool chain can the results be valued correctly during homologation.

Test Environment Adequacy and Fit-for-Purpose Checks

Verification and validation strategies have a full product life cycle impact. Thus, tool environments have to face requirements related to reliability, long-term support, and reproducibility of results. This can only be accomplished if the test environments are used based on a solid quality management system and the recurrent prove of suitability (hardware calibrations).

Risk Assessments, Expert Knowledge, Systems Designs

It has been discussed above that testing for automated driving is strongly interrelated with other development activities. This implies that a test strategy blueprint cannot be a static, stand-alone document. Instead, it needs to adapt to other sources. Such sources are risk assessment methods to be deployed, available expert knowledge on the particular automation system, and also the choice of system architecture and design.

4.2 Data-driven Development and Shifting Responsibilities

How (only) a collaborative approach will enable autonomous driving

Just as software is the key to most of today's automotive breakthroughs, data-driven development is key to establishing new automated driving functions. However, with Big Data, shifting paradigms, and AI, how will players in the automotive industry ensure the safety of functions and the future development of autonomous driving (AD)? A close examination of the situation shows the necessity of the early integration of best practices and comprehensive strategies into the process – a matter of intensive exchange, investment, and strong partnerships.

4.2.1 How does automated driving change the electrical/electronic (E/E) development process?

The standard approach to electrical/electronic development works with automated driving – but only to a limited extent, as the event chains of the driving function are clearly becoming more complex and contain elaborate environmental sensors. Newly established organizations are function-oriented, rather than component-oriented. That means that responsibilities shift, and not just within individual organizations. It also changes the relationship between manufacturers and suppliers, which then, of course, changes development processes and test strategies. At the same time, priorities are shifting as well. Where software algorithms and electronics were previously the sole focus, now AI comes into play. That results in changed process models and the emergence of new roles in data processing. Data train AI, and data are crucial for verifying and validating AI. That requires intensive investments in AI infrastructure in order to develop the right solutions and to support the associated data processes. This encompasses the processing and handling of data, from the acquisition of data logging in vehicles to data enrichment and big data management. Due to the high complexity of systems and their environments the importance of data reaches a new level.

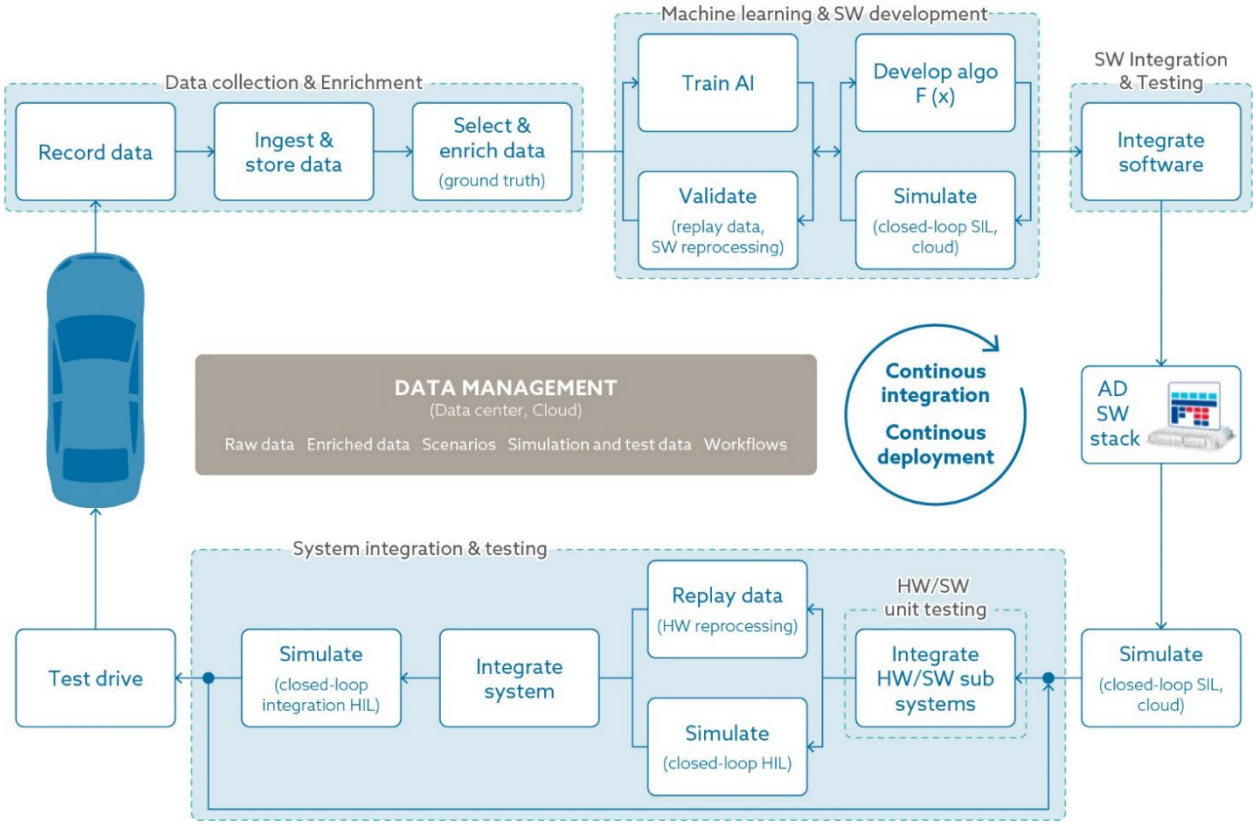
Established methods of model development reach their limits when they encounter data-driven development: AI-learned behavior cannot be modeled directly. A large amount of information must be made available in a systematic way before a sufficiently accurate model can be synthesized.

Thus, AI know-how must be connected with the know-how of electrical/electronic development in the automotive sector. While the introduction of AI is accepted, it cannot lead to weakened competence in development or testing of automotive industry. This is why data-driven development requires a network of partners with an overall view of the process landscape – a partnership that acts independently of domain and thus with a focus on overall development to support the advancement of holistic approaches.

Where previously closed-loop control (prototyping) and validation (closed-loop XIL) were used, data-driven development is enforcing a paradigm shift: data are also being used to stimulate interfaces and subsequent evaluation. This calls for test systems that reproduce recorded and enriched data from test drives or simulations to validate AI algorithms and AI-based ECUs, purely simulatively, based on software or AI algorithms, or directly with the ECU hardware under strict, real-time requirements. Ultimately, testing functional and structural requirements using synthesized and enriched data leads to new challenges in validation. New technologies and methods must be integrated into a holistic process to make this data processing as safe as possible and efficiently connected to all the integration steps along the electrical/electronic development process.

This covers everything from data process to AI training to overall integration and is based on continuously repeated processes that achieve system maturity with increasing amounts of data. Providers of development solutions must, therefore, be involved in the development process much earlier. They will take on the role of development partner to integrate new methods into established processes and to define testing goals and achieve them as efficiently as possible.

Data Driven Development



4.2.2 How does automated driving change the release and homologation process?

Parallel to this shift to data-driven development, as we have established here, the release and homologation of automated driving is also changing. In the past, homologation was the last step during or after development. Random samples were made on vehicles to test overall system behavior. Given the complexity of new systems, this is no longer sufficient. The residual risk of such a highly automated system cannot be determined at the overall system level. In the future, the development process and the (mostly simulative) methods used there will be the basis for the homologation of automated driving functions.

The development of new data-driven development processes should thus also aim to achieve homologation as efficiently and safely as possible, and to fulfill compliance requirements. This means by implication that the requirements of the relevant current and future safety standards must be met.

As such, automated-driving releases will be based on two pillars, largely implemented by simulation: firstly, verification, in which all measures to prevent known risks are checked, and secondly, validation, in which the system is pushed to its limits in relevant use cases to identify unknown risks. Both pillars combine simulation with vehicle testing.

Through combining the available methods of HIL and SIL safeguards, a statement can be made about overall performance and residual risk. The methods will range from classic HIL tests to highly scalable cloud-based simulation solutions that can check an almost infinite number of possible combinations of scenarios. Tool manufacturers must make their products cloud-capable and more flexible than ever before to be able to address the higher number of iterations between verification and validation and the system design. Offerings such as SaaS (safety as a service) might play a role in the future.

The changes and transformations have led as never before to the need to standardize and further explore fundamentals. The complexity arising from software-centric vehicles, the challenges of autonomous driving, and data-driven development can only be managed if standards are applied wherever possible. To make this visible and to put current standards and current research into this context, this needs to be analyzed globally.

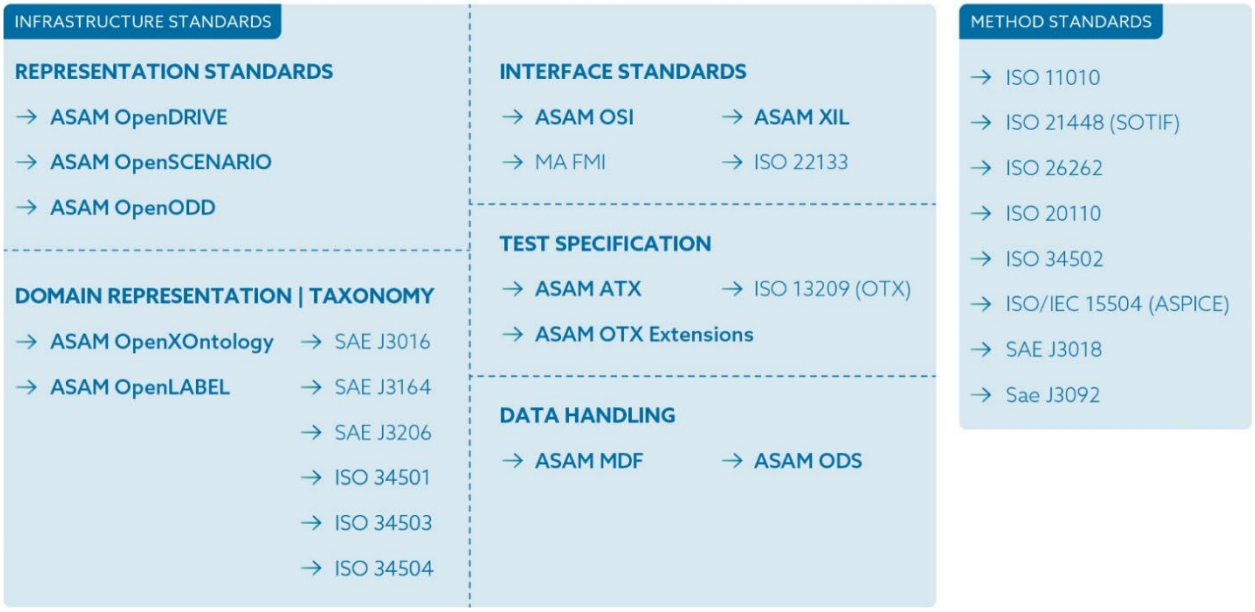
4.3 Existing Standards and Current Research Activities

A 360-degree angle to drive alignment

Analyzing the direction of future test strategies, we need to consider the following: the standardization landscape is not a one-way street. It requires comparing existing standards and comprehensively identifying interrelations, gaps, and phases of development. This report aims to offer insight into the effect of new strategies and current activities, and to ask how the study group's results might affect ongoing efforts. The overall goal: to compile all relevant information and develop a clear-sighted blueprint for testing strategies, adopt advance standardization in the ADAS/AD domain, and provide a common pool of knowledge.

ASAM aims to use existing standards where possible and – if necessary and feasible – to adapt them to the new requirements. Within this study, several of ASAM's industry-proven standards, as well as important standards from other organizations, shall be evaluated for their usefulness for the validation of autonomous vehicles. To structure the analysis of individual standards in the next sub-chapters, this report applies the division of standards as follows:

Overview of Standards for Simulation across Organizations



4.3.1 Infrastructure Representation

These are standards that are used to describe some or all aspects of the simulation or test specific infrastructure.

4.3.1.1 ASAM OpenODD

Description
In order to establish the true capabilities and limitations of automated driving systems (ADSs), we need to first define their operational design domain (ODD). ODD refers to the operating environment (road type, weather conditions, traffic conditions) in which a vehicle can drive safely. For example, for low-speed automated driving (LSAD) systems such as pods and shuttles, the ODD may include urban areas with predefined routes that include pedestrians and cyclists. On the other hand, for a highway chauffeur system, an ODD may include a four-lane divided freeway and dry conditions only. The types of scenarios a vehicle may encounter will be a function of its defined ODD, making this fundamental to any safety evaluation and scenario identification.

A more formal definition of ODD as defined by SAE J3016 (2018) is as follows: “Operating conditions under which a given driving automation system or feature thereof is specifically designed to function, including, but not limited to, environmental, geographical, and time-of-day restrictions, and/or the requisite presence or absence of certain traffic or roadway characteristics.”

In order to enable stakeholders to share, compare, and reuse ODD definitions, there is a need for standards to provide guidance to the stakeholders on both the attributes to be used for the definition of ODD and a format for defining the ODD using those attributes. BSI PAS 1883 (UK) provides a taxonomy for ODD. Additionally, ISO 34503 uses the taxonomy to provide a high-level definition format for ODD. While these standardization activities address the important needs of the industry, a gap still exists in the industry for a definition of ODD adapted for simulation.
ASAM OpenODD is a representation of the abstract ODD specification in a more well-defined syntax and semantics which enables machines to interpret and perform the required analysis.
Additionally, the ASAM OpenODD specification shall be measurable and verifiable for the attributes it specifies.

ASAM OpenODD focuses on a machine-readable format that is:

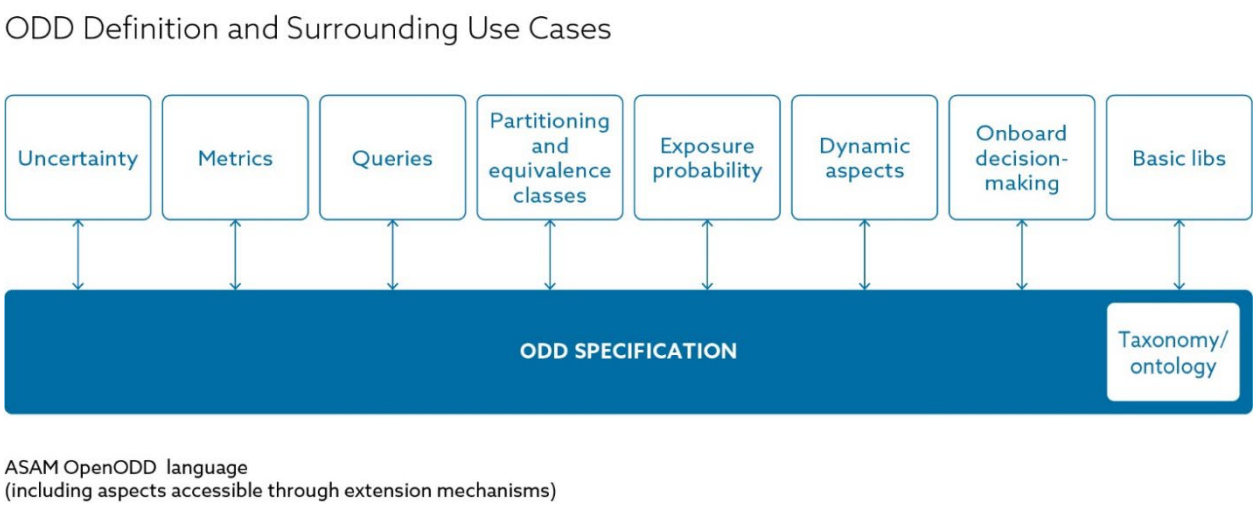
- Searchable
- Exchangeable
- Extensible
- Machine-readable
- Measurable and verifiable

- Human-readable

Current role and relevance with regards to ADAS/AD

In the scenario-based testing workflow ASAM OpenODD will play a very important role supporting the test description, defining the boundaries of what to test, and achieving a good test coverage of the operational design domain and its borders.
It allows for the statistical quantification of:

1. exposure of certain attribute values of the ODD,
2. the performance of a CAV and its systems against the ODD,
3. the tightening or expansion of the ODD over time.



Status

The concept paper was released in November 2021, with a follow-up standardization project starting in March 2022.
View online

Potential implications of the Operational Design Domain and associated testing standards

The study group expects the need to quantify and use ODDs for testing in ADAS/AD to grow significantly over the coming decade, with the need for common ground through, for example, standards, growing proportionately.

The operational design domain is a term that has emerged recently out of the need to more stringently restrict or more clearly delineate the design domain for ADAS/AD functionality. In the past, formal ODDs played more of a minor role in testing, particularly outside the ADAS/AD domain. Their usage was limited to natural-language formulations for on-road testing, with verification or validation against these being limited to the undefined format.

The growing need to be able to specify operational design domain limits in a clear, exchangeable manner is a critical factor in the deployment of AD systems, whose ODDs will presumably increase as the industry advances. Being able to state these limits is, of course, just one facet. As of the authoring date of this report there is but one regulation for a Level 3+ AD system (UNECE 157 ALKS). As regulatory bodies continue making inroads into defining regulations for AD systems and the automotive industry with the development and testing of AD systems, the demand for structured methods and processes to verify against such specifications will soar. This is directly supported by new and upcoming standards such as OpenODD, which provides an underlying exchange format, as well as taxonomy-driven standards like BSI PAS 1883, ISO 34503, or ASAM OpenXOntology.

ODDs will be integrated into a majority of the steps across the testing workflow, from the initial system design and risk assessments to the test specification, where the expected operating limits of a function might be described by an abstract ODD, all the way to being the accompanying artifact for the test case and scenario definition process, with ever-increasing levels of concreteness in the ODD. Validation steps, such as checking scenarios and results against an ODD, will become an integral part of a testing workflow. Test results must begin including ODD criteria, including KPIs to measure degradation of the ODD.

Exactly how ODDs will be integrated into tests remains to be determined and will become clearer as the infrastructure and standards surrounding their definition evolves.

4.3.1.2 ASAM OpenSCENARIO

Description

OpenSCENARIO defines the dynamic content of the world, for example the behavior of traffic participants and how they are expected to interact with each other and the environment. It is used in:

- Virtual development
- Tests
- Validation of functions for:
- Driver assistance
- Automated driving
- Autonomous driving
- Testing on test tracks or proving grounds
- Testing in a mixed environment (HIL)
- Decoding real-world driving data

The OpenSCENARIO standard occupies a unique position amongst the OpenX set of standards in that ASAM currently actively develops two parallel versions of it. The two versions occupy different positions in the application toolchain. V1.0 is a low-level and concrete specification format, primarily designed to be read by simulation tools, whereas V2.0.0 allows those users that create maneuver descriptions and tests to define scenarios at a higher level of abstraction as well as providing alternative expression methods to the current XML format of OpenSCENARIO 1.0.0 via a domain-specific language (DSL).

OpenSCENARIO 2 defines an abstract road syntax that allows for the description of road geometry at varying levels of abstraction. Version 1.1 does not support the description of such static content; instead, it must be referenced through, for example, OpenDRIVE descriptions.

Status

V1.1.0 release date: March 2021
V1.2.0 release date: March 2022
V2.0.0 release date: December 2021

Current role and relevance with regards to ADAS/AD

The complexity of the systems being tested for high levels of automation has (and will continue to) grown exponentially. Whilst the full implications of this are yet to be understood, these levels of complexity need to be reduced at some stage or other in the testing workflow. This can be done at the test specification level (potentially leading to more tests) or at the scenario level. This increasing complexity is directly reflected in the capabilities of the scenario description languages being developed. It is possible that this complexity can be abstracted out significantly better through powerful scenario descriptions that can cover large solution spaces through higher levels of abstraction, which also serve to support increased automation. Further tools to reduce complexity at various points include the robust use of standards as well as ensuring a seamless

exchange of data.

To better understand the target groups, the ASAM OpenSCENARIO 2.0.0 standard documents use cases and users for scenario descriptions in detail. These include various use cases specifically targeting test engineers and regulators (see the ASAM OpenSCENARIO V2.0.0 standard for details).

Whilst only OpenSCENARIO 2 explicitly targets test engineers and regulators, both versions are deemed to be relevant to testing and this report. OpenSCENARIO 1 is currently the only standardized format for scenario descriptions and is hence a key part of current scenario-based testing workflows. This will likely become less so as OpenSCENARIO 2 is released and gains traction in the industry. This is expected to be driven in part by the need to have higher levels of scenario abstraction, for example for regulatory purposes. It is hoped that a standardized format capable of a full spectrum of abstraction levels like OpenSCENARIO 2 will greatly support the use of, as well as the shared understanding of, scenarios in legislature. The roadmap at ASAM for OpenSCENARIO estimates a convergence of the two versions to one in some form within the next three years.

Study group summary of ASAM OpenSCENARIO
As detailed in section 3.5.“The Relation between Test Case and Scenario”, there is a very close interaction between a scenario and a test, which necessitates an interface within the test case description to implement higher-level concepts (e.g. parametrization) consistently in tests and scenarios. This interface must be represented in the scenario as well. It is thus expected that this standard will be extended to support such an interface. Additionally, a scenario needs to have some interface to an ODD description.

Further possible implications include a clearer separation of test-relevant parameters and fields from generic scenario fields, but this is likely to be part of a separate testing-focused standard that can be applied with the language features of OpenSCENARIO. Possibly this could be implemented as a feature set of OpenSCENARIO that is testing-specific. Other such extensions could include bindings to external languages (in addition to the aforementioned testing interface) or testing libraries.

OpenSCENARIO V1 has applied extensive focus in its ongoing development to clarifying the interaction of a scenario description with a scenario engine. Such an engine is then responsible for interacting with a simulation. It is expected that such clarifications will need to be extended to both versions, particularly with an emphasis on how one or more tests activate one or more scenarios.

4.3.1.3 ASAM OpenDRIVE

Description
The OpenDRIVE format provides a common base for describing road networks with Extensible Markup Language (XML) syntax, using the file extension .xodr. The data that is stored in an OpenDRIVE file describes the geometry of roads, lanes, and objects, such as markings on the road, as well as features along the roads, for example signals. The road networks that are described in the OpenDRIVE file can either be synthetic or based on real data.

The main purpose of OpenDRIVE is to provide a road network description that can be fed into simulations for development and validation purposes of AD and ADAS features.

Status
The latest version of OpenDRIVE is V1.7.1, released in June 2021.

Insights into how this standard might evolve
Various use cases leverage road network descriptions without necessarily requiring a scenario. One example is traffic simulations with intelligent traffic agents, which interact directly with the road network without strictly needing a scenario. These use cases will still require some level of interaction between the road network description and the test case or the ODD characteristics. It is plausible to expect that such an interaction could still be realized using the interfaces being defined in a scenario description (see OpenSCENARIO recommendations), with an otherwise empty scenario (no dynamic content like maneuvers). However, it is also possible that a direct need to extend OpenDRIVE with a similar interface to the test and the ODD will evolve.

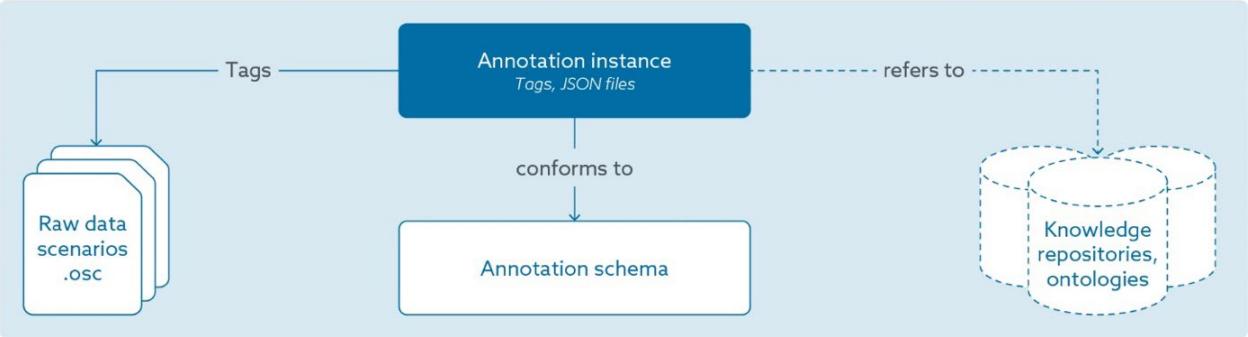
4.3.2 Domain Representation & Taxonomies

These are standards that are used to prescribe taxonomies and ontologies for the relevant domains.

4.3.2.1 ASAM OpenLABEL

Description
OpenLABEL V1.0.0 standardizes the annotation format and labeling methods for objects and scenarios. The use of a standardized format will help save costs and resources in converting annotated data. OpenLABEL will be represented in a JSON format and therefore can be easily parsed by tools and applications. OpenLABEL specifies which coordinate systems are being used as a reference for the label.
OpenLABEL also provides methods to label objects in a scene (one point in time, frame) and over a period of scenes, enhancing this with the labeling of actions, intentions, and relations between objects.

Scenario Tagging Concept



The figure shows the concepts related to data annotation for scenario-tagging. ASAM OpenLABEL covers the definition of the annotation schema for scenario-tagging and an ontology for tags. Scenario-tagging use cases focus on raw data that is used in the development, testing, and validation process of ADAS and AD functions, for example test scenarios or simulation scenarios. Often the format of such raw data is OpenSCENARIO, GEOscenario or other domain-specific languages or formats used to describe and store simulation and test scenarios.

Status
The first version of this standard was released in November 2021.

4.3.2.2 ASAM OpenXOntology

Description

The OpenX standards describe traffic participants, road networks, driving maneuvers, and test scenarios for driving and traffic simulation, labeling sensor data, and other use cases. The standards share concepts from the domain of road traffic, such as roads, lanes, and traffic participants.

OpenXOntology provides an ontology-based architecture for these concepts and thus a common definition of the domain model for the OpenX standards. OpenXOntology covers the following concepts from the domain of road traffic:

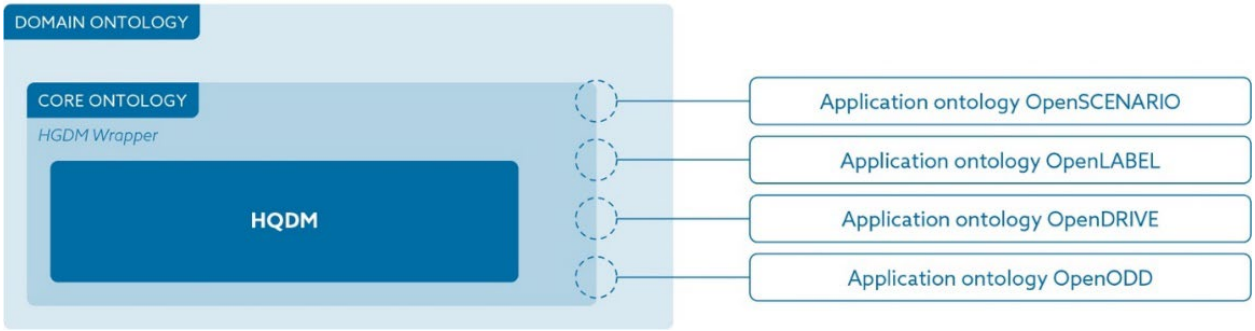
- Road infrastructure, meaning roads and lanes
- Traffic infrastructure, meaning traffic signs, signals, etc.
- Temporal changes in the road and traffic infrastructures, meaning road maintenance, diversions, etc.
- Dynamic traffic agents, such as cars, pedestrians, and cyclists
- Environment, meaning weather, time of day, etc.
- Vehicle-to-everything (V2X) communication objects

ASAM OpenXOntology aims to structure and formalize knowledge about the existence of things in the domain of road traffic. This is done as independently as possible of the intended applications in ASAM standard development so that the ontology can be reused for different types of applications. The current application ontologies are focusing on ODD and scenarios. Nevertheless, the Study Group Test Specification sees the need to investigate the potential benefit of a testing application ontology.

The complexity of creating and using domain-specific ontologies can be reduced by dividing knowledge into multiple areas via modularization. OpenXOntology is divided into the following modules:

- Core ontology – basic concepts such as physical objects, temporal states
- Domain ontology – central concepts of the road traffic domain
- Application ontologies – application- or standard-specific content that is not relevant to the full domain

Example Visualization of Core, Domain, and Application Ontologies



4.3.2.3 ISO/DIS 34501 – Road Vehicles – Terms and Definitions of Test Scenarios for Automated Driving Systems

Description
The standard specifies terms and definitions of test scenarios for automated driving systems (ADSs). The contents are intended to be applied to Level 3 and higher ADSs defined in ISO/SAE 22736. The standard for terms and definitions is the basis for test scenarios for automated driving systems, and the development of appropriate international standards will play a key role in supporting the testing, evaluation, and management of automated driving systems. This standard is used to harmonize and standardize the terminology and definition of test scenarios for automated driving systems on a global scale.
Status
This standard is currently a Draft International Standard (DIS registered, stage 40.40) at ISO.

4.3.2.4 ISO/AWI 34503 – Road vehicles – Taxonomy for Operational Design Domain for Automated Driving Systems

Description
This document specifies the basic requirements for a hierarchical taxonomy for defining the operational design domain (ODD) for an ADS. The ODD includes static and dynamic attributes that can be used to develop test scenarios in which an ADS is designed to operate. This document also defines basic test procedures for attributes of the ODD. This document is applicable to automated driving systems of Level 3 and higher as defined in ISO/SAE 22736. ISO/SAE 22736 also defines the concept of ODD. The definition of ODD is fundamental in ensuring safe operation of the ADS as it defines the operating conditions of the ADS. While different kinds of ADSs may be developed in the industry worldwide, there is a need to provide guidance on a framework for the definition of ODD for manufacturers, operators, end users, and regulators to ensure safe deployment of ADSs. This document will assist manufacturers of ADSs in the incorporation of minimum attributes for the

definition of ODD and allow end users, operators, and regulators to reference a minimum set of attributes for the definition.

Status

This standard is currently in the Preparatory Stage (under development) at ISO.

4.3.2.5 ISO/AWI 34504 – Road Vehicles – Scenario Attributes and Categorization

Description

This document defines packages concerning the different attributes of a scenario that convey information required to construct a complete test scenario. These attributes may refer to complementary data sources, such as the related road network, 3D scenes, and models. To ensure a certain quality standard, measures for the integrity and integrability of these attributes are defined. Additionally, this work item defines a categorization of the scenarios by providing tags that carry qualitative or quantitative information about the scenarios.

Status

This standard is currently in the preparatory stage (under development) at ISO.

4.3.2.6 Potential Implications of Taxonomy-, Labeling-, and Ontology-Based Standards for Testing

Amongst others, this subsection considers:

- OpenXOntology
- OpenLABEL
- ISO 34501
- ISO 34503
- ISO 34504

An underlying requirement for collaboration and data exchange is a shared understanding of the terminology being used. Standardized taxonomies such as those mentioned above provide this baseline in the domain (or a subset of it). The overlap, differences, and relationships amongst the terms being used need to be clear for exchange to be successful. Worth emphasizing is that it is not necessary to have only *one* taxonomy as long as the overlaps and differences in the terms are clear. OpenXOntology currently does this, for example, by clearly mapping terms across the OpenX standards, rather than defining one single taxonomy.

We also recommend the development of an application-specific layer to the OpenXOntology standard for testing that takes all of the conclusions made in this report into consideration. The natural-language-like reasoning enabled by an ontology could also be used to define another layer to scenario and test descriptions that is natural-language-like. This would aim to address, for example, the functional scenario layer when applied to scenario descriptions. Such a layer would enable machine-readable, natural-language-like descriptions in legislature that can easily be converted to more concrete descriptions, thus being less subject to ambiguity or misinterpretation.

The aforementioned taxonomies enable further steps in the testing-specific workflows. A standardized set of tags for scenarios, as defined in OpenLABEL for example, allows a test to specify scenarios based on a specific label. An ontology like OpenXOntology allows for further automation of such a workflow, allowing for scenario selection based on similar or related tags.

This has similar implications for the interaction between an ODD and a scenario database, where tags specified in the ODD can be used to provide the search parameters for matching scenarios.
The OpenLABEL standard with its standardized format for labels for objects of interest (for sensors) is extremely beneficial to testing workflows that leverage different test techniques. Recorded data from a test drive on an open road for example can be used to automate the specification of scenarios for scenario-based testing (see Section 10.2 of the ASAM OpenXOntology standard for more details on this use case).
It can be expected that a standardized set of labels generally eases coverage determination, allowing for the implementation of automata that speed up the process.

Study group summary of this standard

As initial standards and guidelines on terminology for ADAS/AD are published it is expected that significant alignment efforts will be needed to realize the goals and advantages described above. These efforts will need to take place in and amongst the standardization organizations to prevent adding to the confusion in the industry. It is recommended that a coordinated effort to align the different taxonomies is initiated – initially at ASAM internally and in second step across organizations.

4.3.3 Interface Definitions

These are standards that are used to define interfaces or protocols, such as APIs or messaging formats between test relevant models or elements.

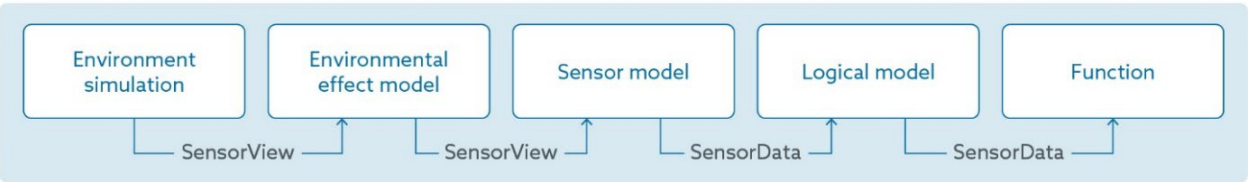
4.3.3.1.1.1.1.1 ASAM OSI (Open Simulation Interface)

Description
To achieve a widespread use of driving simulators for function developers, the connection between the function development framework and the simulation environment has to rely on generic interfaces. ASAM OSI provides easy and straightforward compatibility between automated driving functions and the variety of driving simulation frameworks available. It allows users to connect any sensor, via a standardized interface, to any automated driving function and to any driving simulator tooling. It simplifies integration and thus significantly strengthens the accessibility and usefulness of virtual testing.

ASAM OSI (Open Simulation Interface) started as a generic data exchange interface compliant with the ISO 23150 logic interface for the environmental perception of automated driving functions in virtual scenarios. In tandem with packaging specifications, such as the ASAM OSI Sensor Model Packaging (OSMP) specification, ASAM OSI provides solutions for simulation model data exchange across different implementations.

ASAM OSI contains an object-based environment description using the message format of the protocol buffer library developed and maintained by Google. ASAM OSI defines top-level messages that are used to exchange data between separate models. Top-level messages define the GroundTruth interface, the SensorData interface, and, since ASAM OSI version 3.0.0, the SensorView/Sensor-View configuration interfaces and the FeatureData interface. The GroundTruth interface provides an exact view on the simulated objects in a global coordinate system, the ground truth world coordinate system. The FeatureData interface provides a list of simple features in the reference frame of the respective sensor of a vehicle for environmental perception. It is generated from a GroundTruth message and may serve as input for a sensor model that simulates object detection or feature fusion of multiple sensors.
The following figure shows the interfaces and models involved in modeling a sensor.

Open Simulation Interface Overview

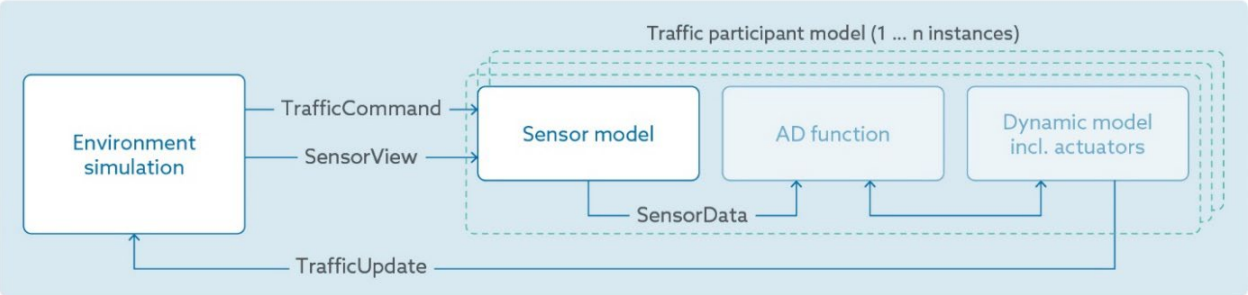


OSI also defines interfaces for traffic participant models. The TrafficCommand interface makes it possible to send commands to traffic participant models. The TrafficUpdate interface makes it possible to receive the updated state from traffic participant models. The following figure shows the interfaces of a generic traffic participant.

Interface of a Traffic Participant

Traffic participant models may use other OSI interfaces internally, for example to model autonomous vehicles. The following figure shows a more advanced use case for traffic participants.

Traffic Participant with Sensor Models, AD Function, and Dynamic Model



Status

The ongoing development activity at ASAM aims to further extend ASAM OSI in order to fulfill the requirement of being able to use it as a standardized interface between further simulation entities.

The latest version of OSI, version 3.4.0, was released in November 2021.

Potential implications of OSI for testing

As a standardized interface for models in a test, including sensor models and traffic participant models, OSI provides significant interchangeability and reusability of models across different test techniques such as HIL, MIL, or SIL. This becomes ever more relevant as multiple techniques are employed to satisfy a test goal. As touched on in the discussion around OpenODD, the systems being tested are ever growing in complexity, and it is becoming increasingly important to reduce the complexity of tests for transparency. Maintaining the same model across tests provides one less variable to take into account.

Further details on the interaction of models with a test are discussed in the section on the Functional Mockup Interface (FMI). The strong interplay between OSI and FMI needs to be taken into account through the development of such testing workflows.

Insights into how this standard might evolve

OSI is continually being developed to support further model types in a simulation or test environment, one example being as an interface to a vehicle-in-the-loop. It is possible that further test-specific or testing-technique-specific messages are implemented in OSI, whether as an extension to existing messages to account for test-specific data or as additional interfaces, for example to an ODD or test layer.

Performance is seen as a limiting factor for many applications of OSI. Prior to extending the current OSI standard further it is recommended that its relatively high performance overhead is improved. Similar considerations have been introduced in the current development project, which aims to investigate alternative, more performant formats to Protobuf.

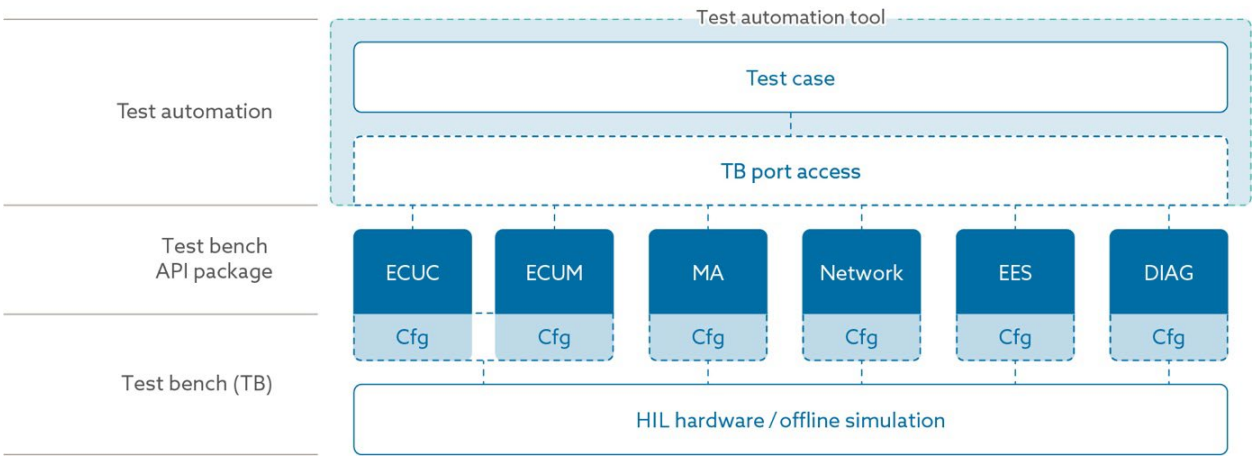
As it is also being developed as an interface between many of the features in the OpenX standards (for example between a scenario engine and a traffic participant or to a traffic signal), any changes made to the other OpenX standards will need to be considered and integrated into OSI. One such change could be extending the TrafficCommand messages to support the exchange of test-specific information with traffic entities.

4.3.3.2 ASAM XIL

Description

The ASAM XIL standard (see https://www.asam.net/standards/detail/xil/ for details) defines interfaces that test automation tools can use to connect to and communicate with test benches. Since the ASAM XIL standard supports not only HIL but also MIL and SIL systems, it can be used in all phases of the development and test cycles. Test automation tools that are ASAM XIL compliant enable interoperability between different test tool and test bench vendors.
The standard provides two levels of abstraction that are built on top of each other: the test bench layer and the framework layer.
The test bench layer defines interfaces for access to different technical areas of a test bench, the so-called test bench ports.

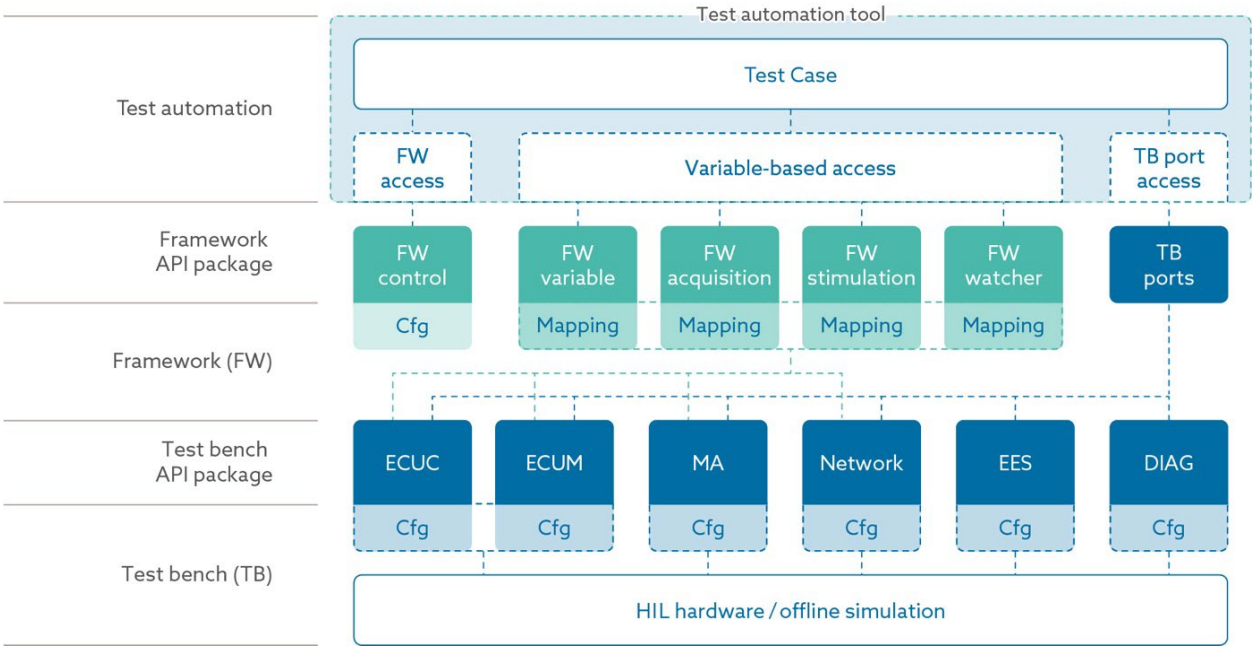
XIL Testbench API with Direct Port Access



Currently available ports are:
The model access port (MAPort) for access to the simulation model. It is possible to read and write parameters, to capture/record signals, and to generate signals.
The ECU measurement port (ECUMPort) and ECU calibration port (ECUCPort) provide access to measurement and calibration tools. They allow the reading and capturing/recording of measurement variables of an ECU as well as the reading and writing of internal calibration values of the ECU.
The network access port (NetworkPort) provides access to network communication, such as frames, PDUs, and signals on CAN or Flexray.
The diagnostic port (DiagPort) communicates with a diagnostic system to read data via diagnostic services from an ECU or functional group.
The electrical error simulation port (EESPort) controls electrical error simulation hardware. It allows different types of electrical errors to be set.

The framework layer is built on top of the test bench ports and provides access to the test bench at a higher level while abstracting more vendor-specific details and configuration options. The purpose of the XIL framework is to achieve effective decoupling between test cases and test benches to improve test reuse.

Test System with XIL Framework API and XIL Testbench API



The XIL framework provides port-independent reading, writing, and capturing of variables, conversion of units and data types, mapping of variable identifiers, and configuration and life cycle management of the test bench ports. Therefore, the XIL framework uses functionality of the XIL test bench ports. The XIL framework also allows the configuration of data acquisition from different data sources, that is, from different XIL test bench ports. The time traces of variables from these different data sources are assembled on a common time basis.
With both XIL test bench and XIL framework, data capturing/recording can be performed based on memory for online processing or based on files into standardized measurement formats (ASAM MDF 4.0 or higher) for data reuse in a later process stage.

References

https://www.asam.net/standards/detail/xil/
https://www.asam.net/standards/detail/xil/wiki/

Status

The standard is well established. The current version is XIL 2.2. Version XIL 3.0 is under development (see https://www.asam.net/project-detail/asam-xil-v300/).
According to ASAM XIL 3.0.0 project page (https://www.asam.net/project-detail/asam-xil-v300/), these are the new features for XIL 3.0 to be developed:

- Introduction of a new value container with generic complex data types
- Introduction of a new test bench port for access to service-oriented communication (SOC): this covers tests of service provider or service consumer as well as integration tests in the network and storage of SOC data in ASAM MDF
- Traceability of test case execution, including logging of events
- Usage of complex setup routines to react to status events of the test system
- Monitoring of variables and states by streaming instead of polling
- Error injection modification to stimulate function signals
- Access to system description via an API
- Rework of the existing ECUM and ECUC test bench ports to one common ECU port
- Usage of the standard in Linux environments

All in all, these are many general improvements, but some have a strong relation to other standards and ADAS/AS development needs.

Current role and relevance with regard to ADAS/AD
Current relations to other standards:
ASAM MDF (Measurement Data Format)
Currently, the XIL standard requires that measurement results be stored in ASAM-MDF-compliant files of version 4.0 or higher.
XIL 3.0 is targeting a solution for the storage of complex data from service-oriented communication in MDF. This may overlap with current MDF activities on the support of image, radar, lidar, and sensor logging (see https://www.asam.net/project-detail/asam-mdf-image-radar-lidar-sensor-logging).

FMI (Functional Mock-up Interface)
XIL relates to the FMI standard in terms of variability of simulation variables.
Furthermore, the growing relevance of (distributed) cosimulation-based test systems raises challenges that are relevant to both standards and may require alignment (esp. regarding system configuration and simulation control).
The relation of FMI to the ASAM XIL standard needs to be evaluated.

Future relations to other standards:
AUTOSAR Adaptive Platform and SOME/IP
The future introduction of an XIL SOC (service-oriented communication) port will eventually have a strong relation to the standards **AUTOSAR Adaptive Platform** and **SOME/IP** (Scalable Service-Oriented Middleware over IP). However, it is expected that the new SOC port interfaces will encapsulate and abstract the technical details of these standards, similar to what is already being done today with the existing XIL test bench ports.

ASAM OSI
The increasing usage of ASAM OSI for environment models (coupling of sensors, agents, etc.) may raise the need for the XIL standard to extend its functionality accordingly to support MIL, SIL, and

HIL testing for future AD applications. Especially the new data types and the binary data type used by OSI may need some adaptions allowing OSI data to be captured, recorded, or stimulated.
In addition, the specifics of configuring and controlling scenario-based environment simulations may raise further requirements of the XIL standard.
The relation of ASAM OSI to the ASAM XIL standard needs to be evaluated.

Additional study group findings on ASAM XIL
The study group has identified interrelations between the standards FMI, ASAM MDF, ASAM OSI, and ASAM XIL as described in this section. The recommendation is to check on the coordination activities between the different standardization projects.

4.3.3.3 FMI – Functional Mock-up Interface

Description
The Functional Mock-up Interface (FMI, https://fmi-standard.org/) is a free standard that defines a container and an interface to exchange dynamic models using a combination of XML files, binaries, and C-code zipped into a single file. It is supported by more than 150 tools and maintained as a Modelica Association Project on GitHub.

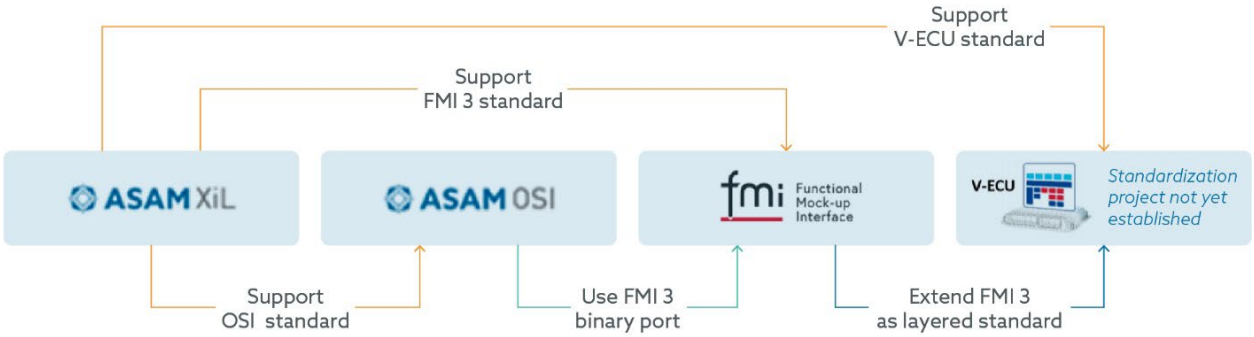
Status
The standard is well established. The current version is FMI 2.0.2. The version FMI 3.0 is under development and will be released soon. Currently, a prerelease beta 1 version is available.
The current version of FMI has several limitations and was released a while ago (FMI 2.0 release in July 2014). Nevertheless, the upcoming version FMI 3.0 will resolve many shortcomings. According to the FMI standard website (https://fmi-standard.org/faq/#what-will-be-the-new-features-of-fmi-30), these are the new features for FMI 3.0 (preliminary):

- Ports and icons
- Array variables
- Clocks and hybrid cosimulation
- Binary data type
- Intermediate variable access
- Source code FMUs
- Numeric variable types
- Extra directory

Current role and relevance in regards to ADAS/AD

All in all, these are general improvements, but some have a strong relation to other standards and ADAS/AD development needs. The figure below shows some of these dependencies.

Direct and Indirect Dependencies of FMI 3 on OSI, XIL, and V-ECU Standards



FMI 3 and OSI

With the FMI 3 binary port, the implementation of OSI interfaces for FMUs (functional mock-up units) will be more robust and replaces an existing work-around (OSMP – OSI Sensor Model Packaging). The binary data type is an opaque binary data type to FMU variables to allow, for instance, the efficient exchange of complex sensor data (OSI) or also field bus data (V-ECU). (Note: The OSI standard itself is discussed in an extra section of this document.)

FMI 3 and V-ECUs

Currently, no standardization project for V-ECUs exists. Definitions for V-ECUs and the need for standardization currently are documented in a PROSTEP white paper: WhitePaper_V-ECU.pdf. The idea is to implement a V-ECU standard as a layered standard on top of FMI 3 to be usable for all kinds of ECU projects (AUTOSAR Classic, AUTOSAR Adaptive, and non-AUTOSAR ECUs).

Besides the FMI 3 binary port for bus data, some more FMI 3 features are important for this layered approach. The FMI 3 “extra directory” adds a new folder in the ZIP archive representing an FMU, providing additional data to travel with the FMU that can be modified by different tools, allowing for layered standards. The FMI 3 “clocks and hybrid cosimulation” feature introduces clocks for synchronization of variable changes across FMUs and allows cosimulation with events.

Thus, the realization of most V-ECU input/output signals with FMI 3.0 will be possible. The following aspects need to be evaluated and then would determine the content for a V-ECU layered standard:

- Bus ports (CAN, LIN, FlexRay, Ethernet):
  Maybe possible via FMI 3 binary ports in combination with MIME types for each bus system, but multiple frames per channel must be possible
- Events and callback functions:
  Events essential for bus communication or simulated hardware events, for instance; callbacks for data measurement services outside of V-ECU; maybe possible via clocks in FMI 3.0

- Address based access to calibration and measurement variables (with A2L files). Not possible with get/set functions, but via direct memory access; extension of FMI especially for V-ECUs necessary
- Simulation performance:
  Events/callbacks via several clocks; if no direct memory access possible, additional memory copies necessary, which reduces performance

FMI 3, OSI, V-ECU, and XIL API

Some new features of FMI 3.0 along with the increasing usage of OSI for environment models (coupling of sensors, agents, etc.) raise the need for the XIL API to extend its functionality accordingly to support MIL, SIL, and HIL testing for future AD applications. Especially the new data types and the binary data type used by OSI need some adaptions to allow them to capture, to record, or to stimulate OSI data. The relation of a V-ECU standard to the XIL API ports (MA, ECUC, ECUM, EES, DIAG, NETWORK) needs to be evaluated. (Note: The XIL API standard itself is discussed in an extra section of this document.)

Study group summary of FMI 3

The findings of the study group do not conflict with the FMI 3 planned features. Nevertheless, the study group has identified the interrelations between the standards FMI 3, OSI, V-ECU, and XIL API as described in this section. The recommendation is to check on the coordination activities between the different standardization projects.

4.3.3.4 ISO/WD 22133-1

ISO/WD 22133 “Road vehicles — Test object monitoring and control for active safety and automated/autonomous vehicle testing — Functional requirements, specifications and communication protocol part 1”

Road Vehicles – Test Object Monitoring and Control for Active Safety and Automated/Autonomous Vehicle Testing – Part 1: Functional Requirements, Specifications, and Communication Protocol

Description
ISO 22133 consists of the following parts, under the general title “Road Vehicles – Test Object Monitoring and Control for Active Safety and Automated/Autonomous Vehicle Testing”:
Part 1: Functional Requirements, Specifications, and Communication Protocol
Part 2: Test Scenario Description Formats

The testing of collision avoidance systems, active safety functions, and more advanced autonomous functions in vehicles requires testing on proving grounds. The purpose is to expose the vehicle under test to potentially dangerous traffic situations in a safe manner. The evaluation is done during development, and in voluntary and mandatory test procedures.
To orchestrate these traffic scenarios, various impactable targets representing traffic actors have to be controlled. The number of controlled targets may be one or many depending on the required traffic scenario. Multiple requirements are important, such as safety, position and speed precision, and logging capabilities.

This document specifies requirements, functionality, and a protocol allowing for multi-vendor target carrier systems to be controlled according to the required traffic scenario, to report expected information for logging purposes and other functions required.

Scope
This document specifies requirements, procedures, and message formats for the controlling and monitoring of test targets, used for the testing of active safety functions and autonomous vehicles. The document specifies functionality and messaging for the monitoring and controlling of test objects by a control center facilitating an interoperable test object environment. This document does not specify the internal architecture of the test object and control center. This document does not specify how the testing of the vehicles shall be performed.

4.3.4 Test Specification

These are standards that are used to define or automate test specifications and their definition.

4.3.4.1 ASAM ATX

Description
The ASAM ATX standard (Automotive Test Exchange Format, see https://www.asam.net/standards/detail/atx/ for details) defines a format for a standardized exchange of test data between different test systems. ATX supports the ISTQB “Certified Tester” syllabus methodology and can be used for many activities in the test process, e.g. test specification, test planning, test execution, and test evaluation.
The ATX test exchange format allows the reuse of existing test cases in different test automation software systems. It enables the description of manual or automated test specifications.

References:
https://www.asam.net/standards/detail/atx/

Supported Activities in the Test Process:

Test Specification
The Test specification activity realizes the general testing objectives by the creation of concrete test conditions and test cases. The major tasks are the following:

- Define and specify the possible test objects
- Define and specify the test base (such as requirements, failure mode and effects analysis, risk analysis reports, architecture, design, interface specifications) and evaluate the testability of the test base and test objects
- Identify and prioritize the test cases based on analysis of test items and the specification, behavior, and structure of the software
- Design high-level test cases (also called logical or abstract test cases) and model the test case flow by defining test steps and test actions
- Document test level, quality criteria, and the used test design technique for the test cases
- Create bidirectional traceability between test base and test cases
- Identify the necessary test parameters to support the configuration of test cases, test steps, or test actions
- Specify the possible test environment setup and document the required infrastructure and test tools

Test Preparation

Test preparation is the activity where test suites are specified by combining the test cases in a particular order and including any other information needed for test execution. Test preparation has the following major tasks:

- Identification and definition of the concrete test object
- Identification and definition of the test environment
- Identification, definition, and/or creation of test data (test parameter values)
- Creation of test execution plans (test suites) from the test cases
- Implementation of unimplemented test actions in the chosen implementation language

Test Execution

In this activity the test environment is set up and the tests are executed. The major tasks are the following:

- Generation of code for the test cases identified by the test execution plan by incorporating the specified test data
- Compilation of the generated code for the test cases
- Setup and verification of the test environment. Optionally, preparation of the test harnesses
- Execution of test procedures either manually or by using test execution tools, according to the sequence in the test execution plan
- Logging of the outcome and the verdict of test execution. Recording the identities and versions of the unit under test, test tools, etc.

Test Evaluation

The test evaluation activity collects the results of a test execution and analyzes the outcome. The major tasks are:

- Comparing results with expected results
- Reporting the differences as incidents in the test report
- Reporting noticeable problems in the test report and analyzing them
- Attaching test logs, result data, etc. to the test report
- Repeating test activities if necessary

Test Environment

A test environment describes the hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test.

Test Object

A test object describes the component or system to be tested. A test object may consist of different parts.

Test Base

A test base represents a chunk of information from which the requirements of a component or

system can be inferred. This might be for example complete documentation, a requirements document, or a single requirement exported out of a requirement management or issue tracking system.

Quality Attributes
There are a number of quality attributes available with which product quality can be described, such as functionality, reliability, usability, efficiency, maintainability, and portability.

Test Design Technique
A test design technique is a method used to derive or select test cases.

Test Specification
A test specification (<TEST-SPEC>) is a container for a set of test cases and a lot of things which they have in common, e.g. the test base, test objects, or test environments.

Test Case
A test case is an essential part of the test specification. It allows the definition of tests and a set of input values, execution preconditions, expected results, and execution postconditions, developed for a particular objective or test condition, such as to exercise a particular program path or to verify compliance with a specific requirement.

Test Step
A test step is part of a specific sequence that serves to divide a test case into steps. They can be documented individually and will show up in the test report.

Test Action
A test action is a single operation inside a test step.

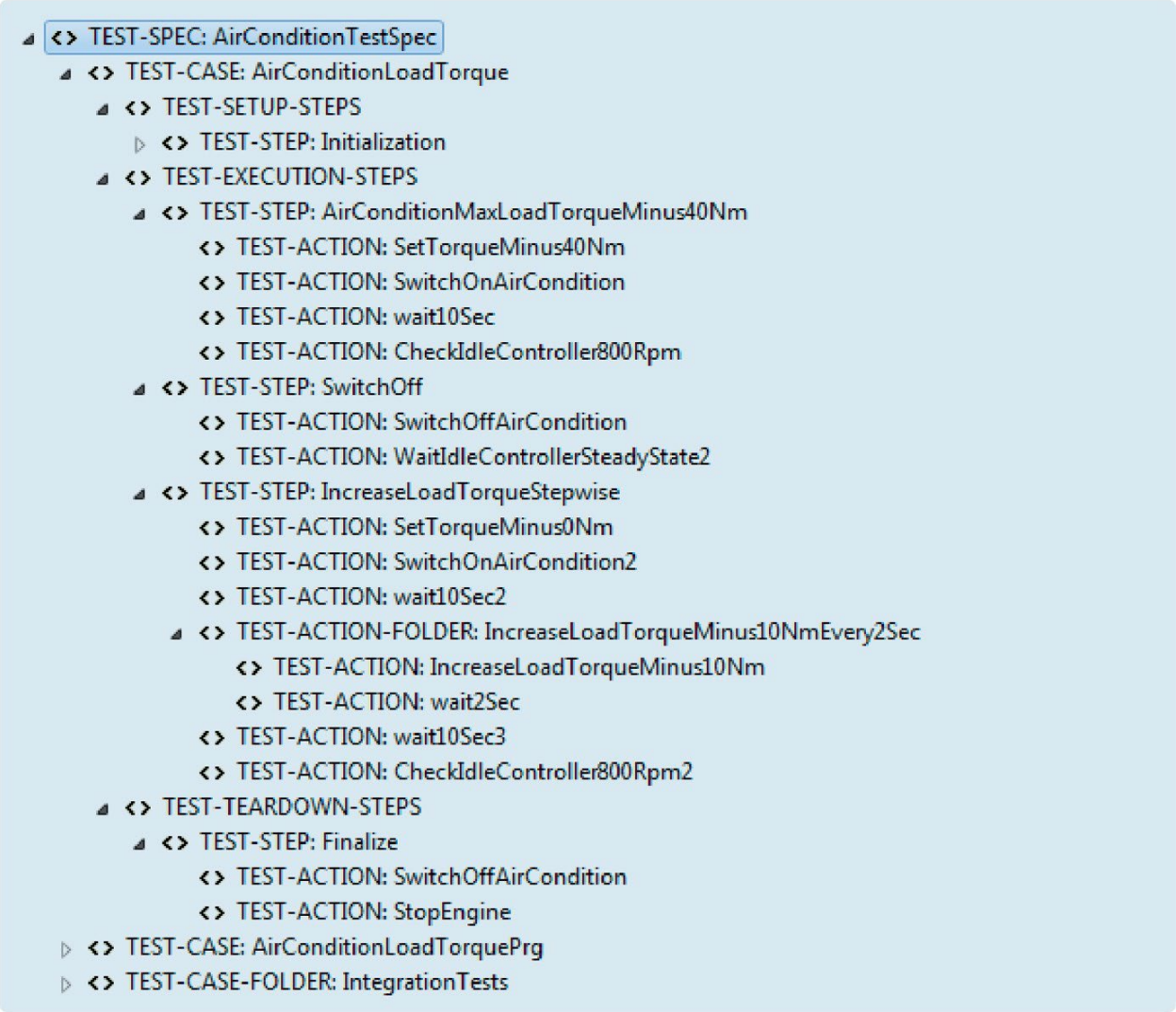
Test Parameter Values
The element <TEST-CASE-PARAM-VALUES> is used to provide the values for test parameters defined within a <TEST-CASE>.

Test Execution Plan
A <TEST-EXECUTION-PLAN> is a test suite describing scope, environment, and schedule of intended test activities. It identifies the test objects, the test environment, the test parameter values, the planned test cases, and their execution sequence. It is a record which can be referenced in a test planning process.
The results of the execution of a test execution plan are recorded in a test report.

Test Report
The test report contains the consequence/outcome of the execution of a test execution plan. For

all executed test cases it includes verdicts and result data like outputs to streams, data changes, logs, or result files.

Example of an ATX Test Case



Relation to other standards

ATX uses the terminology defined by ISTQB. ATX reuses patterns and XML structures defined by the following standards:

- MSR Concepts of Application Profile (MSR-TR-CAP)
- ASAM Harmonized Data Objects (ASAM-HDO)
- ASAM AE Functional Specification Exchange Format (ASAM AE-FSX)
- AUTOSAR V4.0
    - AUTOSAR Generic Structure Template
    - AUTOSAR Software Component Template

By using ASAM XIL API (see section XXX) as interface to test benches/hardware the test exchange flexibility is enhanced due to the abstraction from each specific test bench.

Status

The first version of the ASAM ATX standard 1.0.0 was published in 2012. The current version is 1.0.1 from September 25, 2018. In this release only the TEST-OBJECT-SET was extended. At the current time there are no further development projects.

Source

https://www.asam.net/standards/detail/atx/
https://www.asam.net/standards/detail/atx/older/

Study group summary of ATX

Although the activity is limited, the standard offers interesting aspects in terms of test specifications, metadata, test data, and test plans. However, the standard does not currently support any special features of scenario-based testing. The focus is on classic, requirements-based, and functional testing.

Main benefits of ATX:

- Unified abstract test specification to be shared among different authorities, OEMs, suppliers, and tools
- Development of the test system is independent of test libraries (efficiency increase on end-user side)

ASAM ATX is frequently used in conjunction with ASAM HIL in hardware-in-the-loop test systems.
Source: © ASAM e.V. ASAM ATX, Programmers Guide, Version 1.0.1, Date: 2018-09-25

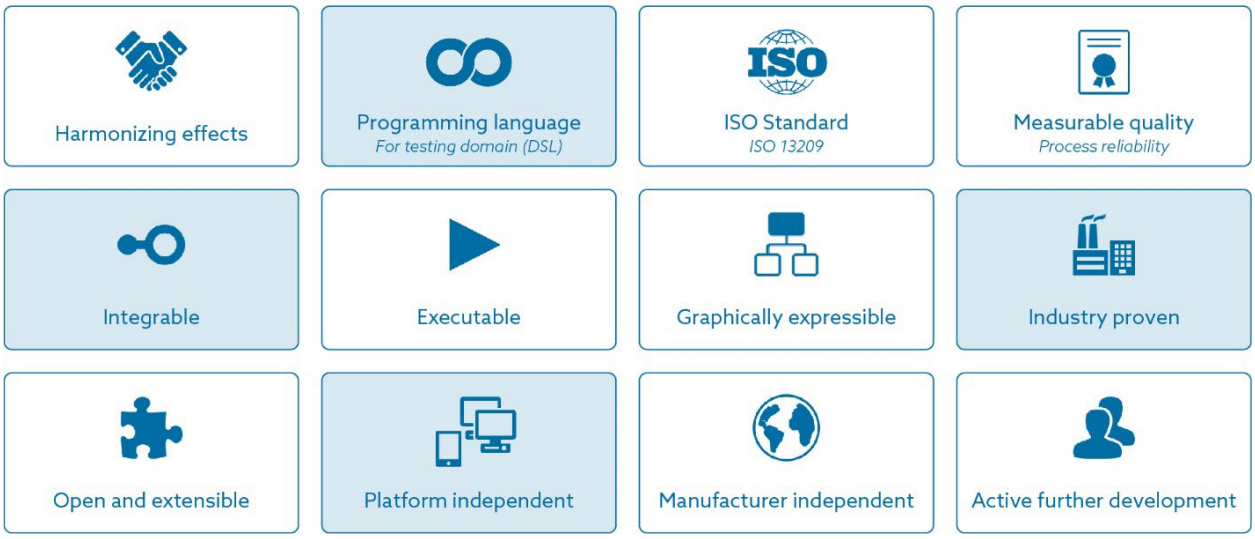
4.3.4.2 OTX (ISO 13209) “Open Test sequence eXchange format”

Description

OTX is a proven industrial standard for test sequences and is as such very important in the field of scenario-based testing. Using OTX, test specification and test logic can be described in a standardized format.

OTX (ISO 13209 and ASAM OTX Extensions)

OTX is first and foremost a programming language, especially related to the domain of testing within the automotive industry. Therefore, OTX is a so-called domain-specific language (DSL). OTX stands for “Open Test Sequence Exchange” and has been an ISO standard since 2012 (ISO 13209). OTX is executable and the test logic can be displayed graphically, where the strict separation of test logic and runtime implementation allows the integration of OTX into any target system (see figure). OTX has harmonizing effects, because it can bring together previously separated standards. It can be extended by any new functionality based on a standardized extension mechanism. OTX has a measurable quality based on about 150 semantic checker rules to improve process reliability. OTX is industry proven, is stable, and is continuously being developed by means of ASAM OTX extensions.



In summary, one can say that

“OTX is a domain-specific programming language for the process-reliable description of exchangeable and executable test logic in the automotive industry.”

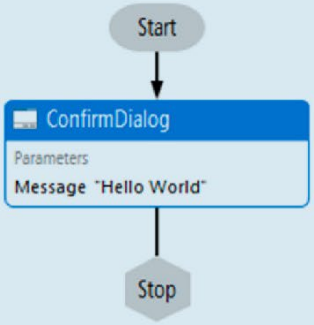
One of the main features of OTX is the standard-compliant exchangeability of executable test logic between development, production, after-sales, and embedded systems inside the car, for example as a so-called on-board tester. This shows that like no other current standard,

“OTX can ensure that the same unchanged test logic can be executed in any target system at any time and comes to the same results.”

OTX consists of a stand-alone executable core (see figure). Inside the core the entire data model for a complete programming language is defined. These are procedure calls, assignments, branches, loops, activities for parallel execution and error handling as well as all extension interfaces which new OTX extensions can implement. Each OTX extension extends the core by means of new domain-specific functionality. This is especially relevant for the vehicle diagnostics domain, but also for the communication between different external systems and for other functionality which is necessary for a complete language, like file access, logging, string handling, date and time handling, and so on.

Existing OTX Extensions (Blue) and Possible New Extensions (White)

Simple OTX "Hello World" Procedure (Displayed Graphically and Textually in OTL Syntax)




```
namespace HelloWorldPackage1
{
    public procedure main()
    {
        HMI.ConfirmDialog("Hello World");
    }
}
```

In the ongoing ASAM project "OTX Extensions" the following improvements have already been implemented or are still being planned (see figure). The core has been expanded by new extension points, so that OTX can support not only the test logic, but also the test description. Based on this a new UnitTest extension was implemented as a prototype (see below for details). A new range extension extends OTX by a set of new simple data types to describe values with validity ranges, for example a float value with an upper and/or lower limit as well as a blacklist and whitelist. Furthermore, OTX can be extended by an ASAM XIL extension to get access to the test bench via the XIL model access port. OTX will also support the upcoming ASAM standard for service-oriented vehicle diagnostics (SOVD) with one or more new OTX extensions. And finally, OTX can be extended for the new ASAM OpenX standards, or for example for the domain of machine learning or big data.

OTX UnitTest Extension

The UnitTest extension is a prototype to show the possibilities of the new extension points. The extension can be used as a base for further extensions. The UnitTest extension adds a new procedure type to the core, the "TestCaseProcedure." A test case procedure contains one or more test cases and can be structured in test suites (see figure for an example).

Test Procedure with Different Test Cases of the OTX UnitTest Extension (in OTL Syntax)

```
[Test]
[TestCase(dividend = 10, divisor = 2, expected quotient = 5)]
[TestCase(dividend = -10, divisor = 2, expected quotient = -5)]
[TestCase(dividend = 10, divisor = 3, expected quotient = 3.33 tolerance 0.01)]
[TestCase(dividend = NaN, divisor = ValueList(0.0,10,NaN,Infinity,-Infinity), expected quotient = NaN)]
[TestCase(dividend = ValueList(10,0), divisor = 0, exception ArithmeticException)]
[Parallelizable]
Divide_FloatFloat(in Float dividend, in Float divisor, out Float quotient = 0.0)
{
    /// Local Declarations

    /// Flow

    quotient = dividend/divisor;
}
```

The simple example of a division of two float values shows some of the main features. The first test case divides 10 by 2 and the expected result is 5. The second test case does the same with minus values. The last test case expects an exception because of the division by zero.
The extension covers about 90 percent of the well-known NUnit testing framework (see NUnit.org), e.g. oneTimeSetupProcedure, setupProcedure, tearDownProcedure, and oneTimeTearDownProcedure.

References

- ISO 13209-1:2011 – Road vehicles – Open Test sequence eXchange format (OTX) – Part 1: General information and use cases, https://www.iso.org/standard/53507.html
- ISO 13209-2:2012 – Road vehicles – Open Test sequence eXchange format (OTX) – Part 2: Core data model specification and requirements, https://www.iso.org/standard/53509.html
- ISO 13209-3:2012 – Road vehicles – Open Test sequence eXchange format (OTX) – Part 3: Standard extensions and requirements, https://www.iso.org/standard/55599.html
- ISO 13209-4:2021 – Road vehicles – Open Test sequence eXchange format (OTX) – Part 4: Expanded extensions interface definition, https://www.iso.org/standard/74090.html
- ISO/DIS 13209-2:2022, https://www.iso.org/standard/76976.html (under development)
- ISO/DIS 13209-3:2022https://www.iso.org/standard/76977.html (under development)
- ASAM OTX extensions, https://www.asam.net/standards/detail/otx-extensions

Relation to Other Standards

OTX has a close relationship to the standards for vehicle diagnostics but it can also be used outside these areas.

- ISO 22900-3:2012 – Road vehicles – Modular vehicle communication interface (MVCI) – Part 3: Diagnostic server application programming interface (D-Server API), https://www.iso.org/standard/56615.html
- ISO 22901-1:2008 – Road vehicles – Open diagnostic data exchange (ODX) – Part 1: Data model specification, https://www.iso.org/standard/41207.html
- ASAM SOVD – Service oriented vehicle diagnostics, https://www.asam.net/project-detail/sovd-service-oriented-vehicle-diagnostics (planned)
- ASAM ATX, https://www.asam.net/standards/detail/atx
- ASAM XIL, https://www.asam.net/standards/detail/xil (planned)

If necessary, OTX can be expanded to support existing standards.

Status

The standard is well established. It is mature and comprehensive enough to replace existing solutions in development, production, and the workshop. Most German vehicle manufacturers have been using the standard productively for years or are planning to use it. We currently estimate around 100,000 installations in development, production, after-sales and within the vehicle, and the trend is growing rapidly. OTX is a process issue. It has a strong influence on optimizing processes.

Numerous well-known tool suppliers already offer a solution or are in the process of developing one.

Example for Scenario-Based Testing with OTX (in OTL Syntax, Taken from the FEV Presentation)

```
[#Name, TestDocument1]
[#Version, 1.0.0.0]
[#Timestamp, 2021-10-02T16:29:52.586+02:00]
namespace ADASTestingSamplePackage1
{
    /// Imports

    import OpenScenario1.DataBase as OpenScenarioDb;
    import OpenScenario1.Model as OpenScenarioModel;


    /// Global Declarations

    [Setup]
    SetupProcedure1()
    {
        /// Local Declarations

        /// Flow

        OpenScenarioDb::SetLanesCount(2);
        OpenScenarioDb::SetRoadRadius(1500.0);
        OpenScenarioDb::SetActiveLane(OpenScenarioDb::ScenarioObject.Lane1);
        OpenScenarioDb::SetDistance(OpenScenarioDb::ScenarioObject.Vehicle1, 1000.0);
        OpenScenarioDb::SetActiveLane(OpenScenarioDb::ScenarioObject.Lane2);
        OpenScenarioDb::SetDistance(OpenScenarioDb::ScenarioObject.Ego, 0.0);
        OpenScenarioDb::SetIgnitionState(true);
        OpenScenarioDb::SetAEBState(true);
        OpenScenarioDb::SetFCWarnTime(5.0);
        OpenScenarioDb::SetActiveGear(3);
        OpenScenarioDb::SetAbsoluteTargetSpeed(OpenScenarioDb::ScenarioObject.Vehicle1, 70.0);
        OpenScenarioDb::SetAbsoluteTargetSpeed(OpenScenarioDb::ScenarioObject.Ego, 20.0);
    }

    [Test]
    [TestCase(positionVehicle1 = 500.0, positionEgo = 100.0, distance = 200.0, warnTime = 5.0)]
    [TestCase(positionVehicle1 = 300.0, positionEgo = 100.0, distance = 200.0, warnTime = 5.0)]
    TestProcedure1(
        in Float positionVehicle1,
        in Float positionEgo,
        in Float distance,
        in Float warnTime
    )
    {
        /// Local Declarations

        Float time2Collsion;

        /// Flow

        if ((positionVehicle1 - positionEgo) < distance)
        {
            OpenScenarioModel::CalculateTime2Collision(out time2Collsion);
            Assertion.Assert(time2Collsion < warnTime, "Test fails");
        }

    }

    [TearDown]
    TearDownProcedure1()
    {
        /// Local Declarations

        /// Flow

        OpenScenarioDb::SetActiveGear(0);
        OpenScenarioDb::SetIgnitionState(false);
    }
}
```

Study group summary of OTX

The OTX standard (ISO and ASAM) is a domain-specific programming language (DSL) for the process-reliable description of exchangeable and executable test logic in the automotive industry. OTX can ensure that the same unchanged test logic can be executed in any target system at any time and comes to the same results. OTX can be expanded to meet the requirements of scenario-based testing. OTX is industry proven and has the necessary maturity.

It is recommended to examine OTX more closely as a description language for scenario-based testing. The creation of a new ASAM working group “OTX Extensions for Scenario-based Testing” is proposed.

4.3.5 Data Handling

These are standards that define formats or processes for data management and storage

4.3.5.1 ODS and MDF

Description

ASAM MDF (Measurement Data Format) is a binary file format for storing recorded or calculated data for postmeasurement processing, off-line evaluation, or long-term storage. ASAM MDF is not only used for the storage of sensor, ECU, bus, or monitoring data, it is also used for the compact storage of large data amounts as external components managed by an ODS (Open Data Services) Mixed-Mode Server.

In addition to the classical bus data like Flexray, LIN, CAN, or Ethernet, new types of data are recorded in the ADAS/AD areas (ground truth data (e.g. GNSS), GigEVision, SerDes/GMSL, FPD Link, or MIPI. These new types of data should be managed and stored in a similar way to the classical bus data in future MDF versions.

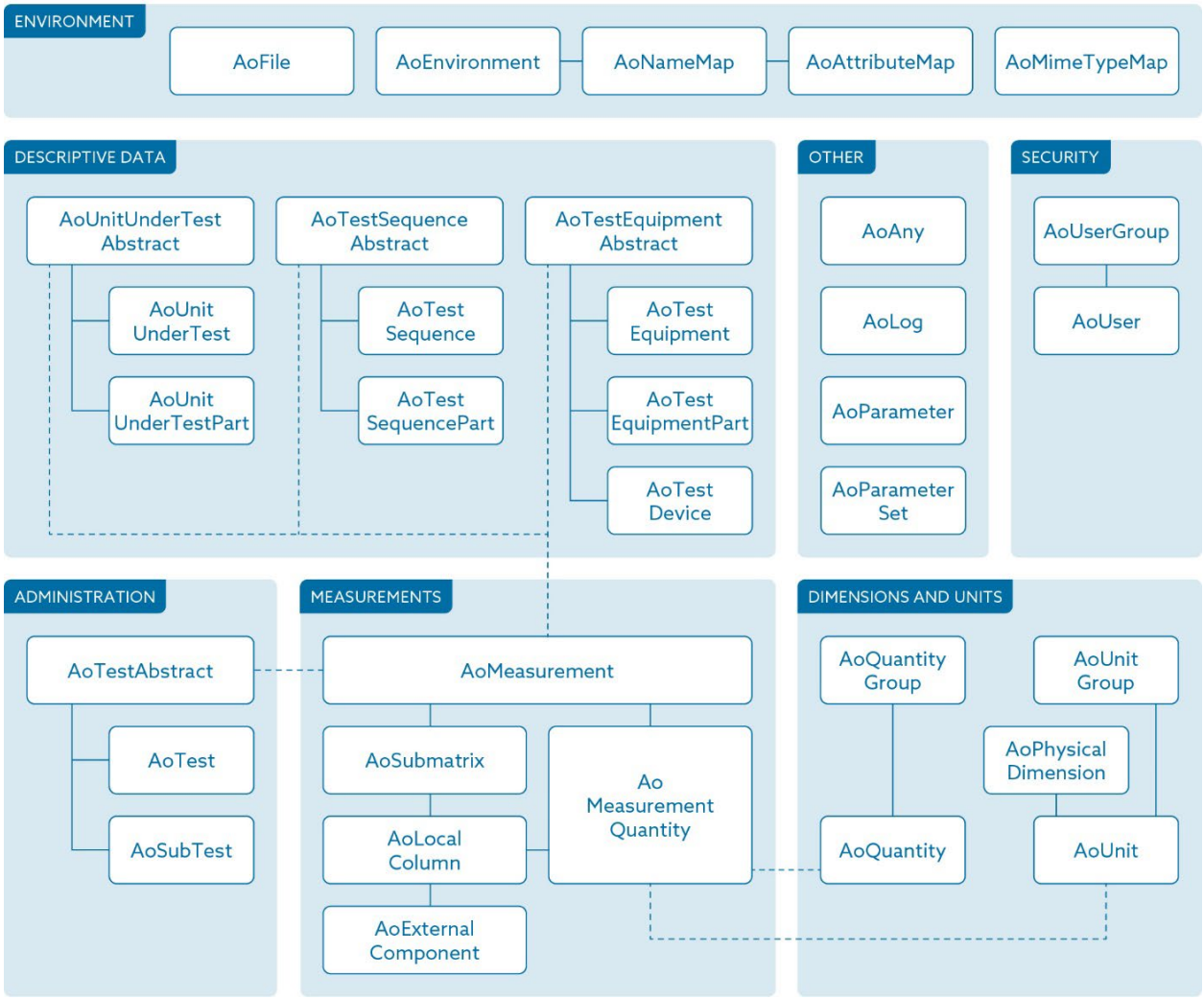
Therefore, ASAM is currently planning to extend the ASAM MDF standard by an associated standard, “Image Radar Lidar Sensor Logging,” that describes how to store image streams that are created in ADAS/AD areas. The aim of this project is to clarify how raw sensor data and stream metadata can be handled in ASAM MDF 4.x and its associated standards.

Results in the testing study group might influence the work of that parallel working group or vice versa.

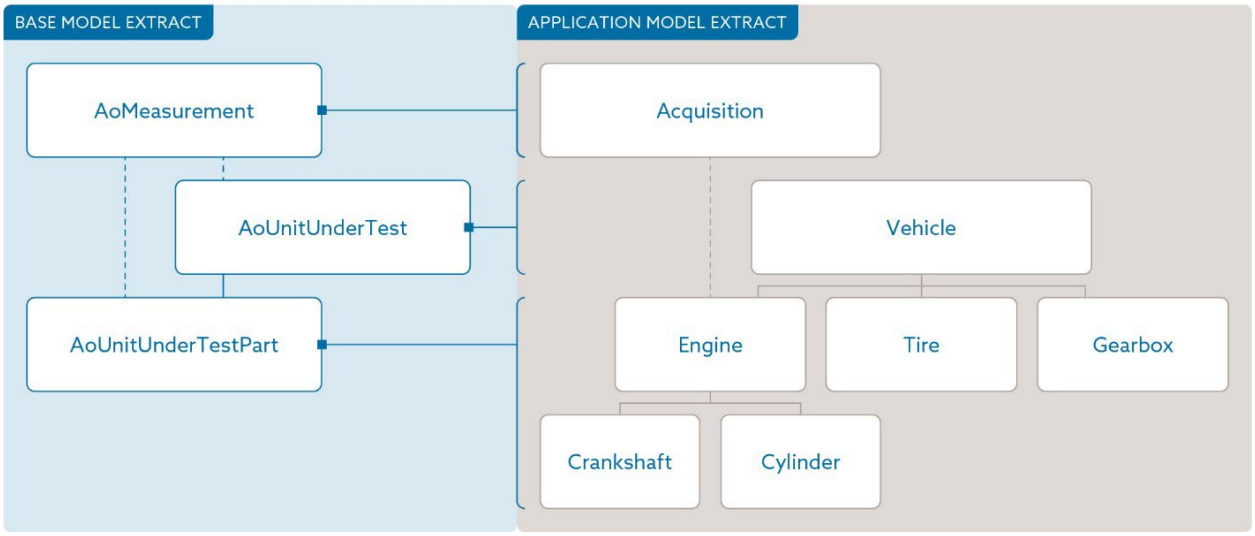
ASAM ODS (Open Data Services) focuses on the persistent storage and retrieval of testing data. The standard is primarily used to set up a test data management system on top of test systems that produce measured or calculated data from testing activities. Tool components of a complex testing system can store data or retrieve data as needed for proper operation of tests or for test data postprocessing and evaluation.

The standard provides the so-called base model. This base model is used to derive specific application models for different application areas. It provides a rough classification of the data in application models by adding semantics to them. This enables client tools from different vendors to correctly interpret the data. Application models can be defined that cover the data storage needs for a variety of application areas.

ASAM ODS Base Model (Source: ASAM ODS 6.1 Standard)



Example Application Model Derived from Base Model (Source: ASAM ODS 6.1 Standard)



Test data saved in the ODS standard can be accessed through standard APIs. For that purpose ODS has released different APIs:

- RPC (90s)
- CORBA API (2003)
- Web-service-based HTTP API (2017)

Benefits of the Standard

The major strength of ODS as compared to non-standardized data storage solutions is that data access is independent of the IT architecture and that the data model of the database is highly adaptable yet still well defined for different application scenarios. Despite this flexibility, clients can query the data from the database and still correctly interpret the meaning of the data.

The ODS standard includes several associate standards, which, for instance, further specify the use of ODS in certain domains, for certain types of test data, or for certain technologies, such as:
Technical driven:

- Storage of ODS data in big-data environments
- Integration of MDF data for storage of measurement data in “mixed mode”
- Storage of bus data
- Storage of calibration data

Domain related:

- Geometry and NVH data
- Workflow specifications (including parallel execution, synchronization, etc.)

Because the storage of MDF files is part of the ASAM ODS standard, the related activities of the MDF working group might have an impact on the further development of the ASAM ODS standard. Following the already-mentioned minor standard “Image Radar Lidar Sensor Logging,” the ODS 6.2.0 minor release will:

- be updated to include the therefore-defined streaming data
- be updated by an interface that allows the integration of user-specific (binary) file formats by different vendors; those could also include non-MDF ADAS test data

Similar to (physical) testing domains, plenty of artifacts are created in the ADAS/AD domains that either describe the execution of tests or test results or contain the recorded data during the tests. Therefore, it seems promising to develop a specific ADAS/AD domain model as an associate ASAM ODS standard. A high-level domain model containing the most important data objects and their constraints has already been identified by the study group (see e.g. chapter 5).

4.3.5.2 SAE J3206

Description
J3206 is an SAE Information Report. This document is nonnormative and imposes no requirements. This SAE Information Report classifies and defines a harmonized set of safety principles intended to be considered by ADS and ADS-equipped vehicle development stakeholders. For automated driving systems (ADS) and ADS-equipped vehicles, there are many interpretations of what constitutes a safety principle. Some principles focus on design and development, some on behavior, and others on maintenance and support of ADS-equipped vehicles. This document is intended to define and develop a set of safety principles for industry use.

These aspects are discussed in detail by the ASAM Test Specification Study Group using the input of some documents referenced in J3206 and others. Therefore, the study group decided not to investigate this standard that has no normative character in a deeper way.

4.3.5.3 IAMTS

Description

The International Alliance for Mobility Testing and Standardization (IAMTS) is a membership-based association of organizations that focus on advanced mobility systems and their testing, standardization, and certification. The purpose of IAMTS is forming a community of global companies, agencies, and organizations in the automotive industry. In parallel, IAMTS supports its members both in obtaining and exchanging information on test environments like proving grounds or simulation platforms and in evaluating them in an unbiased way. Furthermore, IAMTS pushes the creation of a global framework of regulation, test scenarios, validation and certification methods, and terminology.

Status

The International Alliance for Mobility Testing and Standardization has close to 40 top-level members, including industry and innovation leaders, universities, and federal organizations. The organization is currently building a global test environment database to assist OEMs, proving ground operators, and certifiers. Additionally, they announced in May 2021 that they will start working on automotive cybersecurity testing and validation. A recent publication by IAMTS is "*A Comprehensive Approach for the Validation of Virtual Testing Toolchains*" (2021).

Current role and relevance with regard to ADAS/AD

A database where IAMTS collects autonomous vehicle testing data from registered test environments like proving grounds can be of great value to research and development teams working on ADAS/AD functions. Moreover, their network is an opportunity to join the exchange between professionals about key issues of mobility. In addition to that database, IAMTS is also planning to define requirements on scenario databases and virtual testing approaches for the approval of ADAS/AD functions. All combined, this is of great value for testing activities for these functions all over the world and tackles open issues for the safety validation of autonomous vehicles.

Study group findings and suggestions

Comparable to the Study Group, IAMTS has the focus of testing automated driving functions. Even now, there is already a large overlap between the findings of the two activities. It is already clear that virtual testing will play a central role, but more importantly, a combination of virtual and simulative testing and real testing is the key to success. This is a central point for both the Study Group and IAMTS.

Close cooperation between the two activities should be planned. In the view of the Study Group, it is an opportunity for standardization in the automotive sector to agree on the necessary future test standards.

4.3.5.4 ReqIF “Requirements Interchange Format”

Description and background
Requirements management has been an integral part of the development process in various industries (especially in the military, aeronautical, and the medical device industries) for years. Other industries have been adopting requirements management more recently. The automotive industry for example introduced requirements management around 1999.
Now with this established discipline in place, manufacturers and suppliers strive for collaborative requirements management that does not stop at company borders.
For technical and organizational reasons, two companies in the manufacturing industry are rarely able to work on the same requirements repository and sometimes do not work with the same requirements authoring tools. A generic, nonproprietary format for requirement information is required to cross the chasm and to satisfy the urgent industry need for exchanging requirement information between different companies without losing the advantage of requirements management at the organizations’ borders.
With the help of a dedicated interchange format for requirements specifications, it is possible to bridge the gap:

- The collaboration between partner companies is improved by the benefits of applying requirements management methods across company borders
- The partner companies do not have to use the same requirements authoring tool and suppliers do not need to have multiple requirements authoring tools to fulfill the needs of their customers with regard to compatibility
- Within a company, requirement information can be exchanged even if various tools are used to author requirements

The Requirements Interchange Format (ReqIF) specification
(https://www.omg.org/spec/ReqIF/1.2/PDF) defines such an open, nonproprietary exchange format. Requirement information is exchanged by transferring XML documents that comply to the ReqIF format.
The text section above has been taken from section 1.2 of the Requirements Interchange Format (ReqIF) specification.

Status
The current version of the standard is 1.2 from 2016, developed by 14 companies including four OEMs (see https://www.omg.org/spec/ReqIF).
In 2011 the ReqIF Implementor Forum (ReqIF IF, https://www.prostep.org/en/projects/reqif-implementor-forum/) was established to ensure the interoperability of ReqIF exchange tools. The ReqIF Implementor Forum is where tool vendors come together to agree on conventions that go beyond the ReqIF standard, conduct interoperability tests, discuss insights gained “in the field,” and prepare ReqIF benchmarks.

Current role and relevance with regards to ADAS/AD

No autonomous driving-testing-specific aspects of the ReqIF specification have been detected so far.

Study group summary of ReqIF

The basis for deriving the test specification are commonly the requirements of the SUT (system under test). Especially in requirements-based testing the test coverage can be well measured by checking if every requirement has assigned tests. Thus, the requirements are an important artifact within the test data management process which has been analyzed by the study group.
A standardized Requirements Interchange Format supports the transfer of the SUT requirements from the development teams to the test teams, who might work in different departments or companies.

4.3.6 Methods and Processes

These are standards that describe processes or methodologies for testing.

4.3.6.1 ISO 21448 SOTIF

Description
ISO 21448 is about ensuring the safety of the intended functionality (SOTIF). This is referred to as the absence of unreasonable risk due to hazards resulting from functional insufficiencies of the intended functionality or by reasonably foreseeable misuse by persons. ISO 21448 is a necessary extension to global safety standards like ISO 26262 due to the increased introduction of complex technologies like advanced driver-assistance systems (ADAS) or automated driving (AD) to the automotive development process. For these technologies, situational awareness is drawn from a complex interaction of sensors and software. Verifying and validating proper functionality and interaction is therefore safety critical. In this context, the main goal of ISO 21448 is the reduction of residual risk to a reasonable minimum and providing evidence of system behavior within a known environment. The foundation for this is proof that within known scenarios the driving function behaves safely and the likelihood of encountering additional unknown and unsafe scenarios is sufficiently low.
The SOTIF process is a highly iterative process through the three phases design, verification, and validation. Achieving the SOTIF is mainly based on identifying risks that are triggering conditions for system insufficiency. Triggering conditions are specific conditions of a driving scenario that serve as an initiator for a subsequent system reaction, possibly leading to a hazardous event. Based on the system specifications and the operational design domain, a use-cases-based analysis is conducted. The critical triggering conditions that are found in this phase are the basis for system design improvements and the verification phases. To identify the residual risk of the system, ISO 21448 suggests scenario-based tests to validate the system in real-life use cases.
Status
ISO 21448 was published as a publicly available specification (PAS) in 2019. Currently the final ISO 21448 “Road Vehicles – Safety of the Intended Functionality,” which will then replace the current PAS. Because of the industry-wide recognition of the problems that ISO 21448 addresses, there is already significant interest in the standard from many players, with the intention of establishing long-term viable and safe development processes for ADAS/AD functions.

Current role and relevance with regard to ADAS/AD
In contrast to ISO 26262, ISO 21448 does not deal with hazards directly caused by malfunctioning E/E systems (e.g. sensor damage to a camera). Instead, SOTIF focuses on hazards resulting from functional insufficiencies of the intended functionality or from reasonably foreseeable misuse by persons. In regard to ADAS/AD, ISO 21448 is rightfully pushing the need for scenario-based testing, because of the fact that in contrast for example to a steering rod, the complex interactions between sensor and software of an ADAS/AD function cannot be tested in real life. Limiting factors

are not only reproducibility, but also the impossibility to complete an exhaustive execution of all possible scenarios that may occur. The solution to this problem is extensive virtual testing, which is what ISO 21448 is tackling. The standard is therefore closely related to the successful development of ADAS/AD functionalities. SOTIF however does not provide details on how these virtual testing and simulation approaches should be combined with proving ground tests and real driving, although this combination is crucial.

Study group summary of ISO 21448

ISO 21448 is one source for the test strategy blueprint. As the standard is still under development the currently available committee draft is considered. As it also states the combination of several test methods including virtual and simulation-based procedures, this is aligned with the findings in the study group. ISO 21448 will be one pillar for the homologation and therefore it is crucial to consider the requirements when it comes to the release of ADAS/AD functions on the road. The test strategy blueprint defined by the study group is the basis to define an ISO 21448-compliant verification and validation strategy.

As the testing approaches of the study group results and the committee draft of ISO 21448 are already well aligned, it should be considered to reference each other in the final version of the standard to communicate a full view of testing and enable readers to implement such an approach. The test procedures defined in ISO 21448 are a subset of the test strategy blueprint of the study group.

4.3.6.2 ISO 26262

Description

The international safety standard ISO 26262 (“Road Vehicles – Functional Safety”) outlines a framework for functional safety in the automotive industry. The intention of the framework is to establish a uniform, universally accepted approach for the development of safety-related electric/electronic systems. In doing so, ISO 26262 helps to uncover and address possible hazards caused by malfunctions of safety-related E/E systems. ISO 26262 addresses the specific needs of the automotive industry and covers the complete product life cycle, from design and specification, implementation and verification, to production, operation, and maintenance and even end of life and decommissioning of the system.

As ISO 26262 is a risk-based safety standard, a hazard analysis and risk assessment (HARA) forms the foundation for the evaluation of safety-relevant systems. To subsequently identify the safety relevance of a malfunction, the Automotive Safety Integrity Level (ASIL) is determined by combining the frequency of hazardous incidents with their severity and controllability. To complete the holistic framework, ISO 26262 furthermore provides requirements for functional safety management, design, implementation, verification, validation, and confirmation measures in a total of ten normative parts and two guidelines.

Status

ISO 26262 is well established in the Western and Japanese automotive industry. In Asia and especially China, ISO 26262 is becoming increasingly relevant and some companies already implement it in their development process. The first version of ISO 26262 was released in 2011 and covers mass-production passenger vehicles up to 3.5 t. For the second edition of ISO 26262, which was released at the end of 2018, the scope was extended, such that the new edition covers all series-production road vehicles including buses and trucks and motorcycles. Additionally, part 12 of the standard now explicitly addresses an “adaptation for motorcycles.”

Current role and relevance with regard to ADAS/AD

In classical automotive control systems, the type approval is granted based on product testing. This testing is conducted before market introduction. With the introduction of automated vehicle functions, the process of the identification, management, and treatment of risks must be stretched over the entire product life cycle. This results in the need for a continuous application of ISO 262626-compliant testing, verification, and validation processes, to maintain the safe operation of vehicles in an ever-changing environment. Furthermore, moving toward higher automation levels implies that the controllability of malfunctions is reduced. Therefore, extensive verification of safety-relevant functions is necessary.

Study group summary of ISO 26262

The test strategy blueprint defined in the study group is based on various sources. One source is ISO 26262, which recommends a test strategy for electronics and software that is also based on a combination of real and virtual test procedures. The blueprint proposed in the study group, from which the relevant use cases for homologation are derived, is a concretization and extension of the test strategy recommended by ISO 26262. Since ISO 26262 is intended to describe the state of the art for the verification and validation of complex and safety-critical E/E systems and software, we

recommend that an extension and renewal of the proposed test procedures and test strategy be undertaken for the next edition. The blueprint developed by the study group is an excellent basis for this, as it lists all currently relevant test procedures, describes them in detail, and places them in the context of the approval of automated driving functions.

4.3.6.3 ISO 29119 “Software Testing” and ISO JTC1/SC 7 WG 26

Standard: ISO/IEC/IEEE 29119-1 Concepts and Definitions

Description
ISO 29119-1 describes general approaches in software testing

Status
Published 2013, republication expected later in 2021
Current role and relevance with regard to ADAS/AD
Describing test approaches which can be applied in any industry. No specific relationship with ADAS/AD. Typically, industry-related standards (e.g. ISO 26262 family) state requirements on using for example test techniques (such as equivalence partitioning) or processes (such as test planning) without giving detailed advice on how to apply them. This gap can be closed using the ISO 29119 family of standards.

Study group findings on ISO/IEC/IEEE 29119-1
Definitions in this part (or one of the other parts) could be used instead of creating new definitions. Use definitions from those standards whenever possible; if necessary, complement with comments or proposals on how to apply or read definitions in detail

Standard: ISO/IEC/IEEE 29119-2 Test Processes

Description
ISO 29119-2 provides a set of generic processes, one of them on test design, which includes the creation of test cases and test specifications.

Status
Published 2013, republication expected 2021

Current role and relevance with regard to ADAS/AD
Processes can be instantiated to describe scenario-based testing as well as requirements-based testing or other approaches in ADAS/AD.

Study group findings on ISO/IEC/IEEE 29119-2
ISTQB-based process has been referenced; however, this process is coarser than ISO processes.

Recommendation on how to deal with the standard based on the study group (preliminary) results
Refer to ISO process family.

Standard: ISO/IEC/IEEE 29119-3 Test Documentation

Description
ISO 29119-3 provides the content of test specifications, test cases, and other documents created in testing (no details on scenarios)

Status
Published 2013, reissue expected in 2021

Current role and relevance with regard to ADAS/AD
Test specifications and test cases are elaborated and examples given.

Study group findings on ISO/IEC/IEEE 29119-3
Alignment with the terms defined herein is completed

Standard: ISO/IEC/IEEE 29119-4 Test Techniques

Description
ISO 29119-4 provides descriptions of test techniques – but bear in mind that test techniques are understood very differently from the examples in Milestone 0. In ISO 29119, a test technique is a structured way to derive test cases from a test basis. The examples in Milestone 0 would in ISO lingo be test approaches, test levels, or test practices

Status
Published 2015, republication expected 2021

Current role and relevance with regard to ADAS/AD
One of the described techniques is “scenario testing,” which is related to scenario-based testing in automotive.

Study group findings on ISO/IEC/IEEE 29119-4
The concept in the standard is quite generic, but could be referenced. The study group recommends to reference the test technique in the standard and elaborate differences, and explain how this is applied/different in automotive scenario-based testing.

4.3.6.4 ISO/IEC/IEEE 20246 Work Product Reviews

Description
ISO 20246 describes review processes
Status
Published 2017

Current role and relevance with regard to ADAS/AD
Probably not much impact on scenario-based testing; other than reviews are generally beneficial
Relation to the findings of the study group
No relationship

Standard: ISO/IEC 29119-7 Testing of Software for Automotive Systems

Description
ISO 29119-7 is intended to provide mapping between the ISO 29119 family of standards and ISO/SAE 21434, ASPICE, ISO 26262
Status
In preparation
Current role and relevance with regard to ADAS/AD
Might be relevant in the future. Findings of the study group on how this standard might evolve Contribution of ASAM (not necessarily the study group) maybe by official liaison could be useful for alignment.

Standard: ISO/IEC 29119-X TR Test Scenarios

Description
Once this standard is released there will be a direct relationship, although it will have a broader scope (all industries, not restricted to automotive)

Status
In preparation (pre-CD)

Current role and relevance with regard to ADAS/AD
Direct relationship to the study group's work

Study group summary of the standard
Alignment with a future standard is possible. Contribution of ASAM (not necessarily the study group) maybe by official liaison could be useful for alignment.

Standard: ISO/IEC 29119-8 TR Model-Based Testing

Description
The standard is expected to discuss model-based testing (MBT). A scenario could possibly be used as input for MBT

Status
In preparation (pre-CD)

Current role and relevance with regard to ADAS/AD

Only loose relationship

Comparison to the findings in the study group

Using a scenario as input for Model-Based Testing has not yet been discussed. The study group recommends to observe further developments concerning the standard.

4.3.6.5 ISO/DIS 34502 – Road Vehicles – Engineering Framework and Process of Scenario-Based Safety Evaluation

Description

This document provides guidance and a state-of-the-art engineering framework for ADS test scenarios and scenario-based safety evaluation processes. The engineering framework clarifies the overall scenario-based safety evaluation process to apply during product development. The guidance and framework are intended to be applied to Level 3 and higher AD systems defined in ISO/SAE 22736.

Status

This standard is currently a Draft International Standard (DIS registered, stage 40.40) at ISO.

4.3.7 Research Projects – Ongoing or recent

With different focuses and project structures, the following research activities currently deal with testing and developing tools and strategies for the validation and verification of autonomous driving functions. There are benefits for both the study group and the research project: standards are valuable input for research projects, often used when combining different software and hardware components provided by individual partners. Additionally, research projects aim to boost the development of not only individual companies, but of the automotive industry in its entirety, which is best achieved by considering solutions with a maximum of standardization. During research projects, important problems in the field of autonomous driving are investigated, and innovative solutions are developed. Research projects are thus very interesting for standard development, hinting at future development in the field. As one can see from the following summaries of current research activities, scenario-based testing and validation plays a key role in shaping the future of autonomous driving, especially when combined with AI algorithms. Here, we provide some insights into exemplary projects with high relation to testing.

4.3.7.1 SETLevel

Project duration: 2019 – August 2022
Funded by the German Federal Ministry for Economic Affairs and Energy (BMWi).

SETLevel ("Simulationsbasiertes Entwickeln und Testen von automatisiertem Fahren" – simulation-based development and testing of automated driving) tackles the problem that it is not possible to ensure the absolute traffic safety of automated vehicles based on real driving tests only.
SETLevel deals with the simulation-based development and testing of automated vehicles. In the project, research and development staff of 20 project partners in science and industry are cooperating to form the basis for reliable verification methods and thus for later approval of automated driving functions.
SETLevel builds upon the PEGASUS cooperation project completed in May 2019. In that project, partners from science and industry developed quality standards and methods for making autonomous vehicles safer. The focus was on the context of motorway traffic. SETLevel develops the simulation approaches of PEGASUS on a broader basis and extends the application to the complete traffic environment.

SETLevel states that the scenario-based analysis and testing is essential for automated driving. An overall concept of managing all relevant modules for that purpose has been developed (model interface architecture based on Open Simulation Interface) and is used to guide the project. Requirements from application situations are considered and evaluated with specific simulation use cases. The Credible Simulation Process (https://setlevel.de/assets/forschungsergebnisse/Credible-Simulation-Process.pdf), embedded in the simulation-based engineering task and model creation process, has been refined and is seen as an important reference process to bring the required traceability to scenario-based simulation and testing.
SETLevel complements the work of this study group very well. The three selected simulation use cases test existing and developing standards (OpenScenario 1.x, OpenDrive 1.x, OSI, etc.) and feed the findings back to standards development. By connecting research institutes, tool vendors, and OEMs/tiers, practical needs of productive application projects, possibilities of new methods and research implementations are brought together and assessed in an iterative manner.

4.3.7.2 V&V Methods (VVM): Verification and Validation Methods for Automated Vehicles in Urban Environments

Project duration: 2019 – June 2023
Funded by the German Federal Ministry for Economic Affairs and Energy (BMWi).
SETLevel and VVM build on the results of the PEGASUS project, which was concerned with identifying and describing critical scenarios and turning them into universally applicable test cases for highly automated vehicles, using the Autobahn Pilot as an example. The Verification and Validation Methods project, or VVM for short, extends the PEGASUS method, taking a downtown junction as an example.
The approach mapped out for the VVM project can be described in three essential steps. Validating an autonomous vehicle by means of test drives in normal road traffic would require driving several million kilometers in order to cover enough different situations. So VVM's first job is to investigate the combinations of effects that lead to critical situations in urban road traffic. To cover the event space, existing databases and expert knowledge are used, in addition to dedicated simulations that are being developed. As a result, the test space can be reduced to scenarios that are actually relevant, making the amount of effort involved in tests more manageable.
The second step will be to use the findings to develop a safety concept and a functional concept for describing automated systems, which can also be applied to hierarchical subsystems and components. This will make it possible, in the future, to validate new components in independent tests – instead of performing labor-intensive real test drives as is currently the case. In the final step, an example implementation of the validation framework will be produced as a demonstration. The goal is to produce an integrated, dynamic test environment in which the various test platforms can be combined flexibly from simulation right through to real drives, while at the same time allowing for an evaluation of the overall safety level. This will mean that tests can be moved systematically from the real world to simulation, resulting in greater time and cost efficiency.

Related publications:

Westhofen et al., "Criticality Metrics for Automated Driving: A Review and Suitability Analysis of the State of the Art," https://arxiv.org/abs/2108.02403.
Neurohr et al., "Criticality Analysis for the Verification and Validation of Automated Vehicles," in IEEE Access, vol. 9, pp. 18016–18041, 2021, doi: 10.1109/ACCESS.2021.3053159.
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9330510

4.3.7.3 SmartLoad

Project duration: 2018 – October 2021
Funded by the German Federal Ministry of Education and Research (BMBF)

The relevance of SmartLoad for this report lies in its focus on testing and validation in the development of ADAS functionality, especially through a study on the usability of the OTX standard for scenario-based testing.

In the SmartLoad research project new development and test methods for failure prediction and prevention are developed to ensure safety and reliability for individual components as well as for the autonomous electrical vehicle itself. The results are evaluated with reference applications from the areas of interurban individual mobility, commercial vehicles, and public transport.
One way to ensure reliability is the usage of redundant components, which increases the costs of the vehicle. SmartLoad develops new concepts for a functional safe design by optimal and cost-efficient usage of redundant components and functions, with simultaneous intelligent condition monitoring and damage prediction.
In SmartLoad the new development process uses modular and standardized methods to prove the influence of interactions between automation functions and damage mechanisms on the reliability of the vehicle and its components. The elements of the development process are a scenario-based simulation of the driving and working task of the vehicle, the use of damage models in reference architecture, and test execution on distributed and connected component test beds. The new test methods and processes are based on the ASAM standards OpenDRIVE, OpenSCENARIO, and OTX.
In the scope of the project the specification and execution of a scenario-based test using OTX and OSC 1.0 for the test and scenario description conducted in collaboration with RWTH Aachen with a positive result. The results of this study were presented as input for the ASAM Test Specification Study Group and did contribute to the discussions on the role of the standard OTX for scenario-based testing.
The joint project is coordinated by AVL Deutschland GmbH, with Forschungszentrum Informatik, IEW (University of Stuttgart), IPEK and FAST of the Karlsruhe Institute of Technology, Schaeffler Technologies AG & Co. KG, and RA Consulting GmbH participating as partners.

SMARTLOAD
Neue Methoden zur Zuverlässigkeitssteigerung
von hochautomatisierten elektrischen Fahrzeugen

Bundesministerium
für Bildung
und Forschung

4.3.7.4 KISSME

Project duration: January 2021 – December 2023
Funded by the German Federal Ministry for Economic Affairs and Energy (BMWi)

In regard to the report and the study group results KIsSME is deemed relevant because of its close relation to open-road testing.
The KIsSME [1] ("Artificial intelligence for selective near-real-time recordings of scenario and maneuver data in testing highly automated vehicles") project investigates how artificial intelligence can reduce the amount of data generated during the development of ADAS functionality.
Especially during the testing of highly automated vehicles, large amounts of data are generated. The KIsSME project aims to reduce this by developing an AI-based method that recognizes critical driving scenarios. Only if the current driving scenario is considered critical is the corresponding data recorded. This means recording only those data fragments during driving that add value for development and evaluation.
This approach will reduce and condense the accumulating data volumes to save storage space, electricity, and evaluation effort. The recorded data of relevance will be used off-board for rebuilding ASAM-standard-conform scenarios which can be used for the further development of automated driving functions, in terms of training, validation, and simulation.
The joint project is coordinated by AVL Deutschland GmbH, with Fraunhofer Institute for High-Speed Dynamics, Ernst-Mach-Institute, Forschungszentrum Informatik, Karlsruhe Institute of Technology, LiangDao GmbH, Mindmotiv GmbH, RA Consulting GmbH, and Robert Bosch GmbH participating as partners.
ASAM e.V. () as well as Cluster Electric Mobility South-West / e-mobil BW GmbH, State Agency for New Mobility Solutions and Automotive Baden-Württemberg are acting as associated partners.

[1] KIsSME stands for "**K**ünstliche **I**ntelligenz zur **s**elektiven echt-zeitnahen Aufnahme von **S**zenarien- und **M**anöverdaten bei der **E**rprobung von hoch-automatisierten Fahrzeugen."

5 A Blueprint for New ADAS/AD Test Strategies

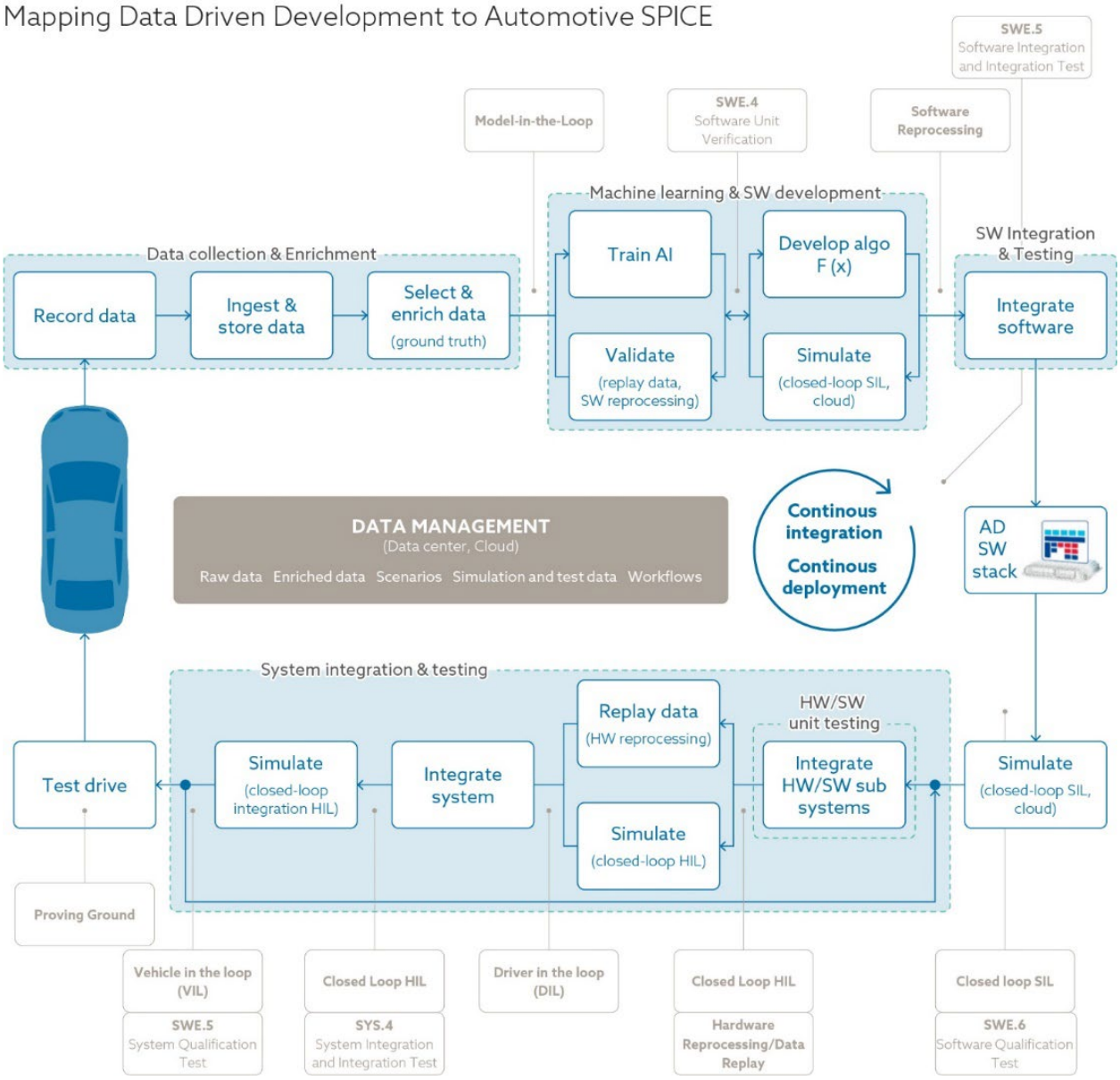
ASAM Study Group Results and Testing Methods

ADAS features are designed for safe and error-free driving over several hundred thousand miles. But while the demand is growing, so is the complexity of testing requirements. The changing vehicle architecture, along with software-centricity and the necessity of sensing the environment in an open world, increases testing challenges exponentially.

5.1 The Derived Blueprint

Tools, testing, and timing

Current test strategies, which are often heterogeneous and have grown over several vehicle generations, must be viewed holistically and used for vehicle approval. Safety argumentation can only be provided for these software-centric and (partially) autonomous vehicles if testing, verification, and validation are holistically managed, defined, and assessed.
The blueprint shown should be seen as a starting point, enabling the automotive industry to challenge its own established procedures, to adapt them, if necessary, and to consciously use them to ensure safety. Nowhere else offers such a holistic overview. Holism is the core aspect here: right from the early phases of testing, through to the open road, test procedures must be designed to be consistent, analyzable, evaluable, and comparable in order to meet current challenges and fulfill future requirements.



This blueprint is based on current safety standards, established best practices, and important norms. If you look at the data-driven development process previously introduced, it quickly becomes clear that although this brings new requirements with it, it by no means completely undermines a general approach.
If you take Automotive SPICE, for example, and combine the phases of testing required there with data-driven development, it quickly becomes clear that the same requirements also come into play here. We do not have to reinvent the wheel, but the wheel must evolve.

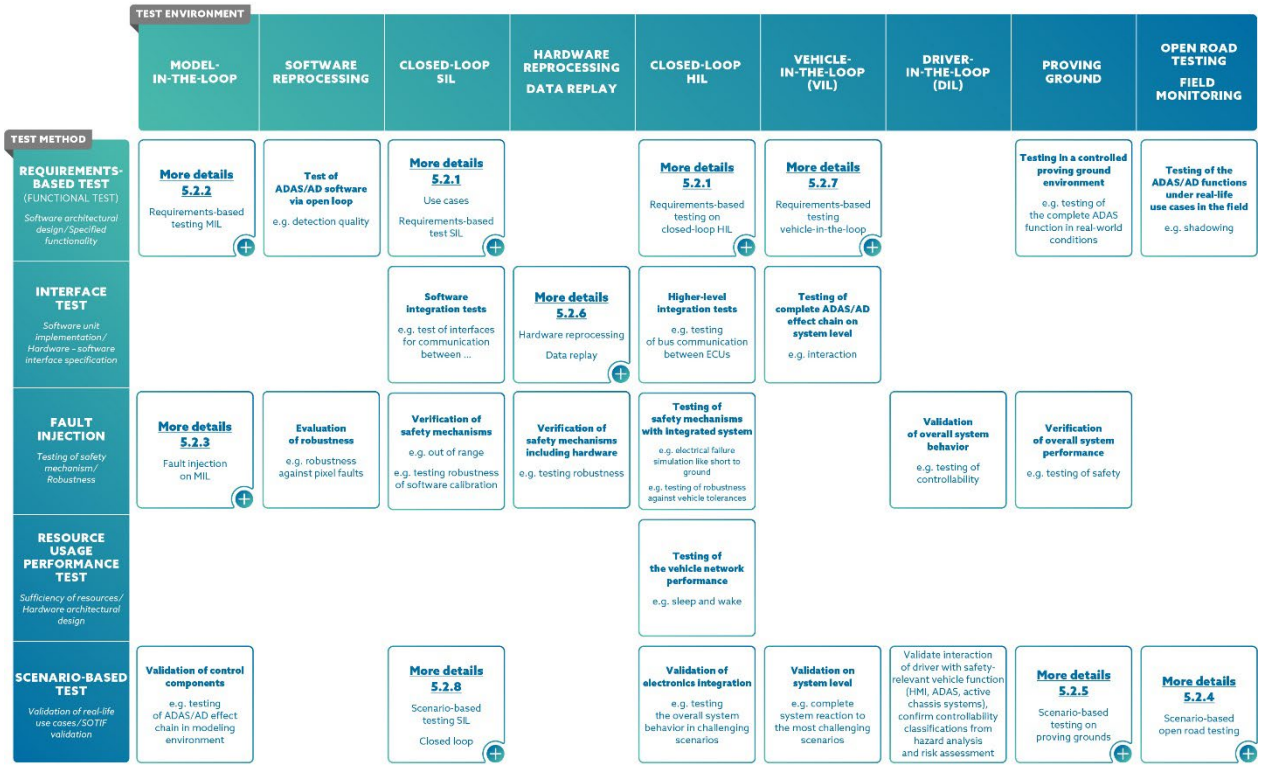
The various test procedures highlighted in the blueprint can also be clearly anchored in the data-driven process. It is important to understand that the phases are no longer strictly separate from one another, but that transitions are smooth and continuous. Iterations and changes between the

phases occur constantly and merge into one another. Accordingly, continuity, a procedure that is as standardized as possible, is unavoidable. In order to be able to approve the vehicles at all, there is ultimately no alternative to this holistic approach.

Testing Matrix/Overview

For the creation of the blueprint, we use a classical approach in test strategy development. The necessary test environments are defined in relation to the test methods required for the safety argumentation. The meaningful and efficient combination of test environments and test methods is then further specified and described in more detail in the form of use cases and workflows.
Here, you can see a possible and reasonable combination to fulfill a test coverage for a software-centric and (partially) autonomous vehicle, which is sufficient for release and homologation. The combinations shown here are one possibility; we have described them in more detail in the following sections. In general, it is clear that the individual phases cannot be considered separately, as holistic safety argumentation relies equally on all pillars. The use of meaningful synergies and transitions between test methods and test environments is essential for vehicle safety.
With the complexity of today's vehicles and the enormous importance of electronics and software, it is no longer possible to do without this type of holistic test strategy.

Test Methods and Use Cases



5.2 User Journeys and Use Cases

Keep it in the loop, accelerate, or (let it) hit the brake

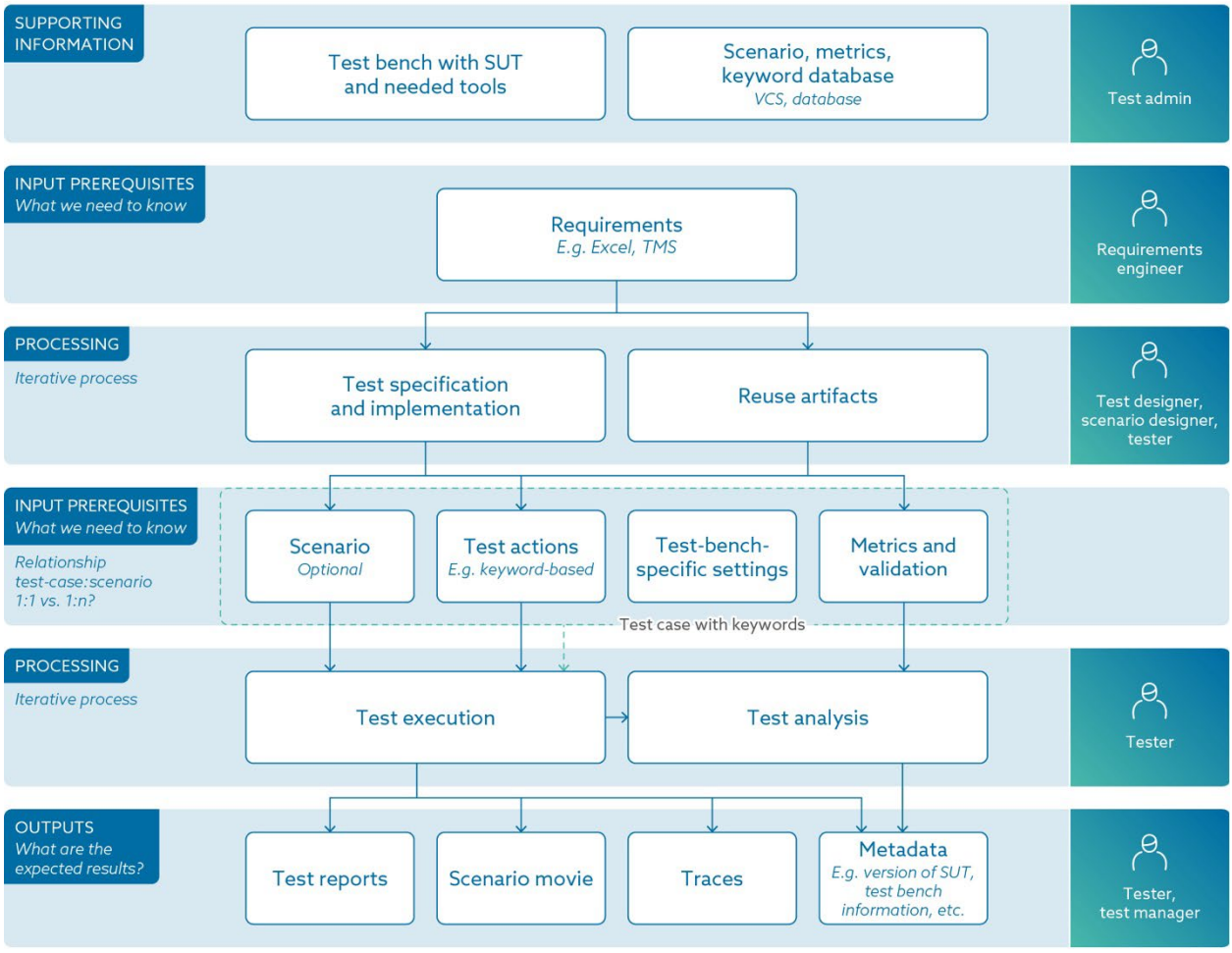
In order to find a sophisticated approach and overcome the emerging obstacles, every building block needs to contribute to a seamless workflow. Still, how test cases and scenarios engage within the user journey can vary widely.

5.2.1 Requirements Based Testing on Closed Loop HIL + Requirements Based Test SIL

This section presents requirements-based testing with scenarios in the area of hardware in the loop (HIL) and software in the loop (SIL). The following user journey demonstrates the high-level view of the workflow.
The focus of the examples is on the interaction between test cases, scenarios, metrics, and test-environment-specific conditions.

User Journey
In this exemplary user journey, the different phases for the validation of ADAS/AD functions in the area of requirements-based testing are shown. Here, the focus is on integration tests for HIL, SIL, and possibly also model in the loop (MIL) platforms:

Testing of an Automated Emergency Braking System in a HIL/SIL Environment



Tools involved: environment access (bus, diagnostic, fault infection),
device under test access, test automation, requirement tool, scenario editor

Requirements are the starting point for testing driving functions. The test designer creates the test specification according to the requirements. These contain both the information about the test sequence
Based on the test specification, all necessary building blocks for the test execution are created or already existing artifacts are reused. This includes the scenario (optional), the test case, metrics, and test-bench-specific preconditions to be created.
Ideally, test executions are triggered by the availability of corresponding new versions of the driving functions. The results include the test report with the procedure of the test case, records of scenario executions, further records (e.g. bus communication), and further metadata (e.g. software data, hardware data, additional test data, etc.) for traceability.

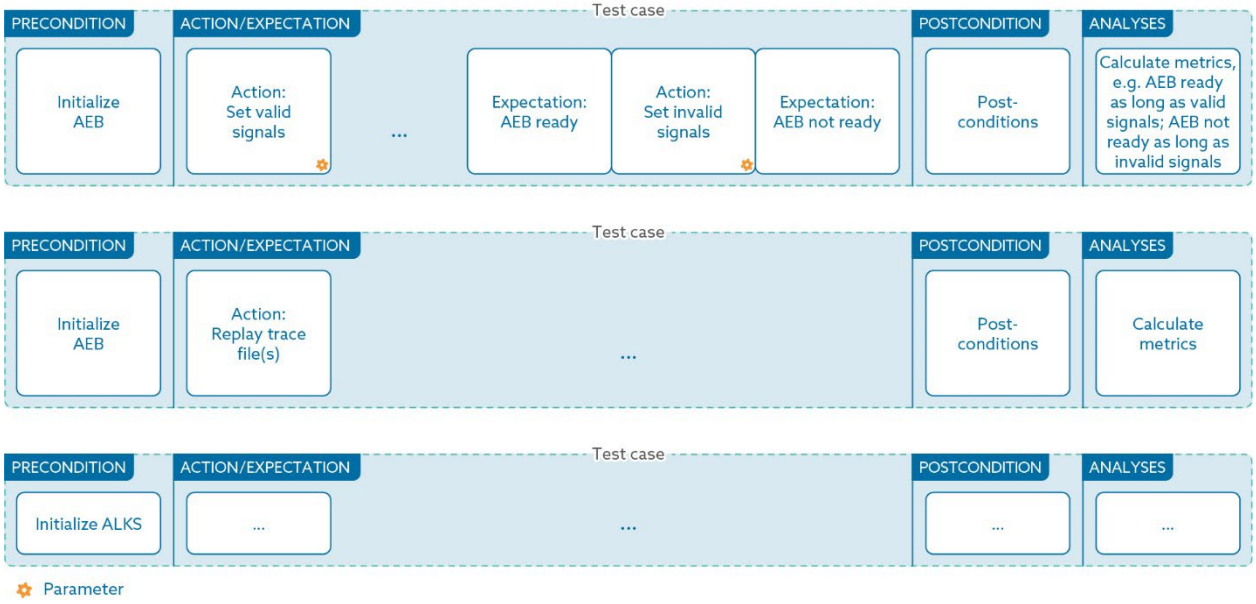
Requirements concerning Test Specifications

Requirement	Evaluation
Requires scenario	SIL: Optional, depends on SUT HIL: Optional, depends on SUT (example ESP-ECU)
Requires hardware	SIL: Hardware for SIL environment (e.g. cloud server) HIL: SUT, Restbus, and test bench definition and configuration required
Requires coordination of tools and models	Optional (toolchain orchestration)
Requires exchangeability among test instances	Test criteria, models, scenarios, reports
Configuration of stubs/drivers/mock-up	optional
Kind of interface between test and scenario	Optional, depends on the relationship between test case and scenario (see examples)
Covered by best practice, regulations, and standards	n.a.

Focus of Use Case 4:

- Testing of driving functions without scenarios, e.g. open-loop component testing in the field of SIL/MIL

Use Case: Testing ADAS/AD Functions AEB, ALKS, etc.
Specifics: Testing without Scenarios



Conclusion

The use cases are structurally very similar in HIL and SIL. Reuse of test cases and scenarios is possible, but not trivial, since specific test environment steps are quickly built into the artifacts unconsciously. A keyword-based approach can mitigate this problem by distinguishing between the general test specification (keyword-based) and the implementation. Depending on the platform (SIL, MIL, HIL), the corresponding implementations of the keyword are assigned.

5.2.2 Requirements Based Testing MIL

This user journey describes a requirements-based generic model-in-the-loop (MIL) testing process as applicable in model-based development processes. In this context it is irrelevant whether the model artifacts are used for the generation of productive controller code or merely for the purpose of simulation. In both cases a fault-free model is required for further development steps to be based upon.

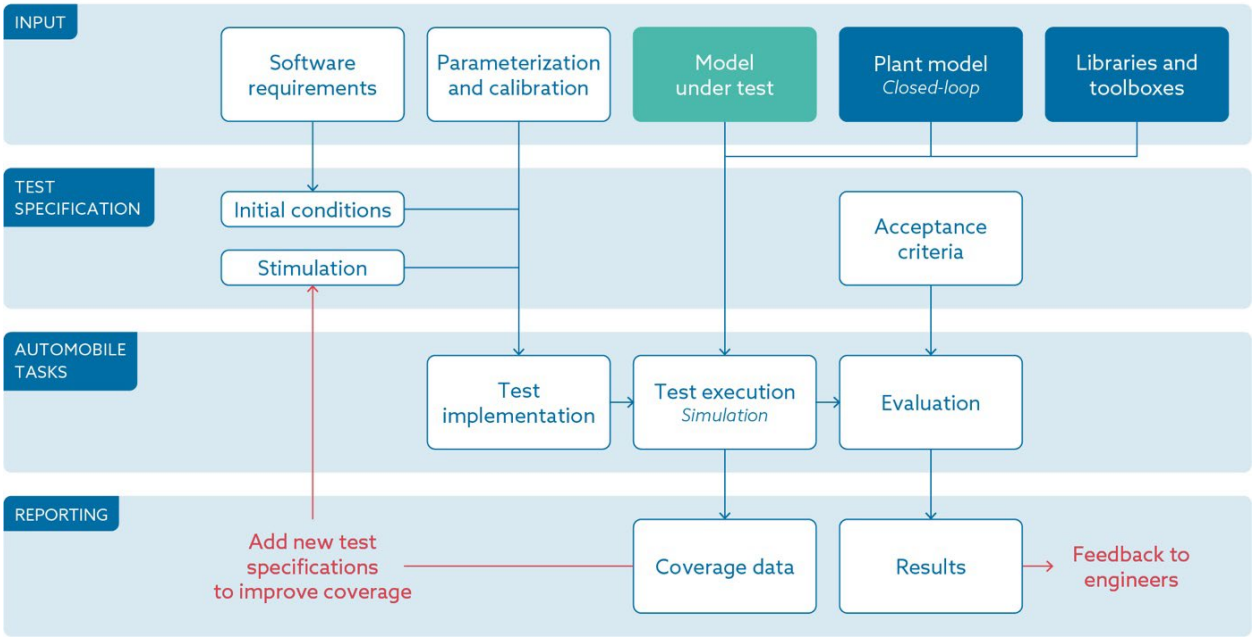
Motivation

MIL testing is most effective if applied closely after or even at the time of active model development. In contrast to hardware-in-the-loop (HIL) testing, which rather serves the purpose of verifying a system's function as comprehensively as possible, MIL testing is available much earlier in the development process and comes at significantly less cost. Thus, it is intended to find flaws inside a (potentially still incomplete) functional model and instantly provide the documentation for remedy, rather than proving the accuracy of a fully finished system.

User Journey

In requirements-based testing, tests are often derived directly from the requirements to the system under test.

Executing a MIL test always involves the simulation of the model under test on an appropriate simulator, usually a part of the modeling tool, the test tool, or a standalone tool, often equipped with supplementary libraries and toolboxes. For the test implementation, however, there are two options. It can be part of the simulation along with the model or it runs externally while communicating with the simulator and the model via the harness. In the latter case, the test tool is responsible for the synchronization of both components. This is possible if it has full control over the simulation time, i.e. it can pause/resume simulation or even execute it stepwise and hence perfectly synchronize the test script with the model. During the execution, the simulator or the harness are responsible for capturing relevant output data of the model. Capturing and storing the entire input data also absolves the test tool and the test implementation of the liability of correct stimulation ex-post, so it is recommended. In the context of white-box testing, the *model coverage*, i.e. the ratio of executed model parts over their total number, aggregated over the entire test suite, is a good metric of its comprehensiveness. When it is too low – 100% coverage is a frequent claim – either further tests must be added to the suite or dead parts within the model should be eliminated.



The test engineer can inspect the captured data in accordance with the previously specified acceptance criteria. Either this is done manually or the test tool performs the evaluation automatically, creating a test result and report. If the report contains relevant findings of a mismatch between the test specification and the behavior of the model observed during simulation, it must be assumed that the model does not meet the requirements which the test specification has been derived from. Thus, failed test reports should be fed back to the departments who are responsible for model specification and development.

In many cases, traceability of the original requirements throughout all test artifacts is desired. That means that for each test (case), test implementation, test execution, and test result it is known which requirement(s) they validate against and vice versa.

Example: Test on Adaptive Cruise Control

Consider a simplified adaptive cruise control (ACC) system as part of the ADAS functionality of a modern vehicle. One of its requirements shall be "*At a speed of more than 70 km/h, when the preceding vehicle is more than 25 km/h slower than the ego vehicle and less than 80 m ahead, the ACC triggers a warning sound.*"

The ACC system is developed in a model-based fashion where productive controller code is auto-generated from a functional model and linked against an auxiliary, handwritten library. The ACC model has input ports for the current ego speed (V_{ego}) and for the distance to the preceding vehicle (d_{pred}). It provides an output for the aforementioned warning sound.

The test engineer derives an open-loop test specification from the above requirement. In order to provoke the ACC's sound trigger, a drop of d_{pred} below 80 m at a rate of at least 25 km/h ≈ 6.94 m/s is required. He/she formulates the following specification that comprises a drop of 10 m/s:

- Stimulate V_{ego} with a constant value of X km/h, where X is calibratable.

- Stimulate d_{pred} with a constant value of 90 m for 2 seconds. Then ramp down to 70 m within 2 seconds and keep that value for a further 2 seconds.

The expected output depends on the calibration. For X > 70 km/h, the sound trigger must rise from *false* to *true* after exactly 3 seconds. That is when d_{pred} falls below the 80 m threshold. For X ≤ 70 km/h, the trigger must stay *false* during the entire test. Further initial conditions are not required in this simple scenario. Three calibrations are specified: {X=100}, {X=60}, and {X=0}.
The test harness is automatically created for the given model under test. The test tool derives one test script for each calibration, executes the test script, and evaluates the results against the specified expected behavior. The generated report contains the actual data that has been used for stimulation, so its validity only depends on the soundness of the simulator.
If the expected behavior deviates from the captured results, the report can be passed back to the model developer, who can trace back the failure with the data. Otherwise, it can be archived.
With proper tooling, MIL testing allows the simulation of the model under test embedded into an environment or plant model with less effort than with software-in-the-loop (SIL) or HIL testing.
In the above example of an ACC controller, such an environment model would also contain and thus simulate the remaining components of the ego car. Thus it could, for example, calculate d_{pred} based on the ego speed, which in turn is derived from the ACC's output values for brakes and throttle, and feed this back to the ACC's input. This way, the indirect influence of the outputs of the model under test on its own stimulation can be tested and evaluated.

Requirements concerning Test Specifications
The table below lists some requirements and characteristics of the presented example to allow a quick comparison to other user journeys based on common criteria.

Requirement	Evaluation
Requires scenario	Optional
Requires hardware	Simulation platform (usually a PC, server or computing cluster)
Requires coordination of tools and models	Yes: Test-harness creation and feedback to evaluation algorithms (usually both automated)
Requires exchangeability among test instances	Not required but beneficial
Configuration of stubs/drivers/mock-up	In large parts automatable/derivable from the model
Kind of interface between test and scenario	Only indirectly via system requirements that manifest themselves in a specific scenario

5.2.3 Fault Injection Testing MIL

This user journey describes how fault injection testing can already be applied at the model-in-the-loop (MIL) level and in which contexts and regards it can supplement the classic fault injection testing activities performed at the hardware level.

Motivation

Fault-injection testing serves mainly two purposes: First, it checks whether functionalities that are intended to be implemented in a fault-tolerant manner indeed sustain the fault. Second, it analyzes the behavior of not-fault-tolerant functionalities in case of failure due to a specific fault. Some measures are implemented in hardware, e.g. via redundancy, and therefore cannot be tested other than through hardware-in-the-loop (HIL) tests. For those measures that are part of the functional model, however, MIL testing is not only available but offers significant benefits when performed in advance of the HIL test.

As for requirements-based model-in-the-loop testing, fault injection testing at the model level intends to find flaws in a functional model. This distinguishes it from later testing activities such as particularly hardware-in-the-loop testing, which are carried out at a stage where the system under test is already assumed or even required to be free of errors. The main reason is that MIL testing takes place very shortly after or even during model development. Hence, feedback cycles are relatively short and error-correction costs are much lower when compared to HIL testing. Furthermore, fault-injection testing is very effective and highly automatable at the MIL level, allowing a system to be tested against hundreds of faults per minute, creating perfectly reproducible results.

There are limitations of course. For example, MIL testing is not able to simulate faults in components which are not part of the environment model, such as sensor failure. In these cases, only their expected impact on the model can be simulated, which can be sufficient for test-driven development but is not for proper system validation. Likewise, hardware robustness tests that involve the application of physical stress to the devices can hardly be carried out at the MIL level.

User Journey

As explained above, MIL fault-injection testing is well-suited to check and evaluate the behavior of a functional model in case of a fault, mainly to check its reaction to the fault. Measures such as fallback modes or even fault-tolerant controls are often realized in software and already part of the functional model the code is derived or generated from. In these cases, what is nominally entitled as “fault” is nothing more than another *valid* input for the model under test. A different scenario is a fault that occurs *inside* the model under test. Both types of faults, external and internal ones, can be simulated and evaluated with appropriate MIL test tools.

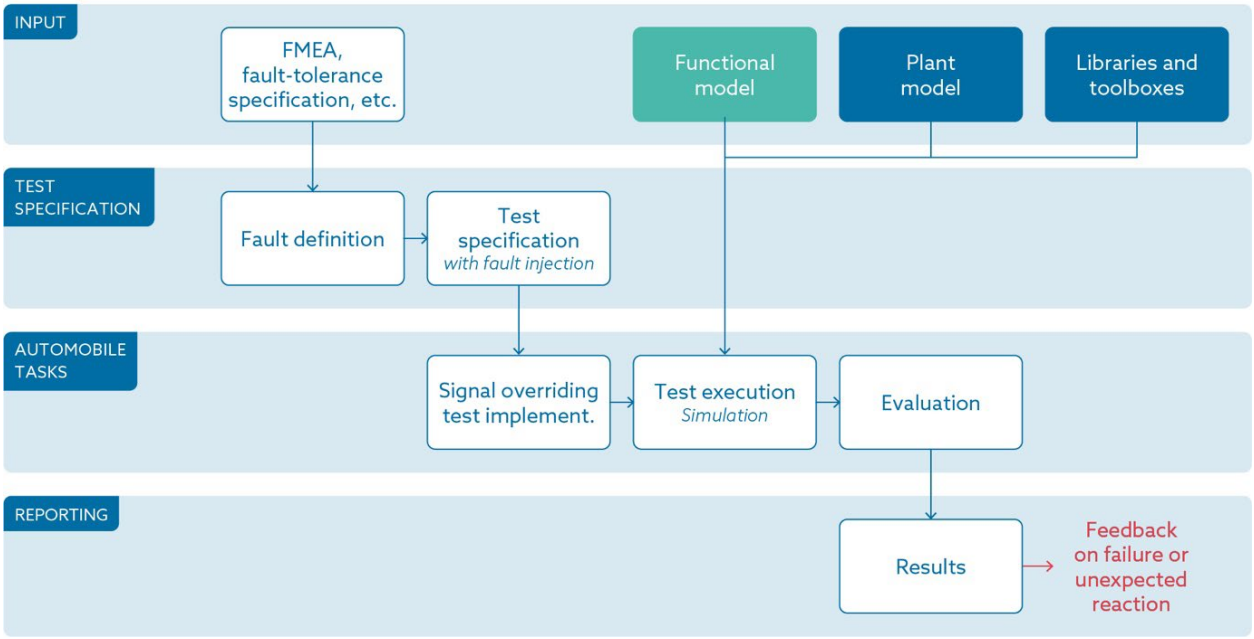
Usually, the basis of fault-injection testing is given by a document that files and specifies different types of faults along with the effects they may, must, and must not have on the functional system. From this documentation, the test engineer derives semiformal definitions of specific faults he/she wants to test the system against. These can be external or internal faults or even combinations of both kinds. Based on the definitions, concrete test specifications can be derived. These often

follow a generic pattern:

1. Initialize the system and drive it to a certain state via regular stimulation
2. Inject a fault
    a. to the environment model *or*
    b. to the stimulation of the model under test *or*
    c. into the model under test
3. Continue stimulation and capture the system's behavior under the fault

Besides the stimulation, the specification should contain the expected reaction to the fault in terms of expected internal and output signals.
Fault-injection testing is often applied in a closed-loop manner, i.e. the input ports of the system under test are stimulated by an environment model that in return consumes its outputs. In most cases, neither the environment model nor the model under test provides any fault-injection mechanisms. The test tool thus needs to locate the variables/signals where the fault shall be injected and *override* them with the specified faulty value. In the above cases of a. and c., the tool must provide capabilities to not only override the functional model's inputs but any arbitrary signal inside both models' hierarchies.
During simulation, all relevant data must be captured. These encompass especially the (regular) stimulation, all variables/signals with which a fault is injected, and all output signals and all internal signals that give relevant insights into the behavior of the system under the fault.
As for requirements-based MIL testing, an automated evaluation of the recorded signals against the expected signals can be performed in the post-processing step, followed by the extraction/generation of conveniently readable test reports.

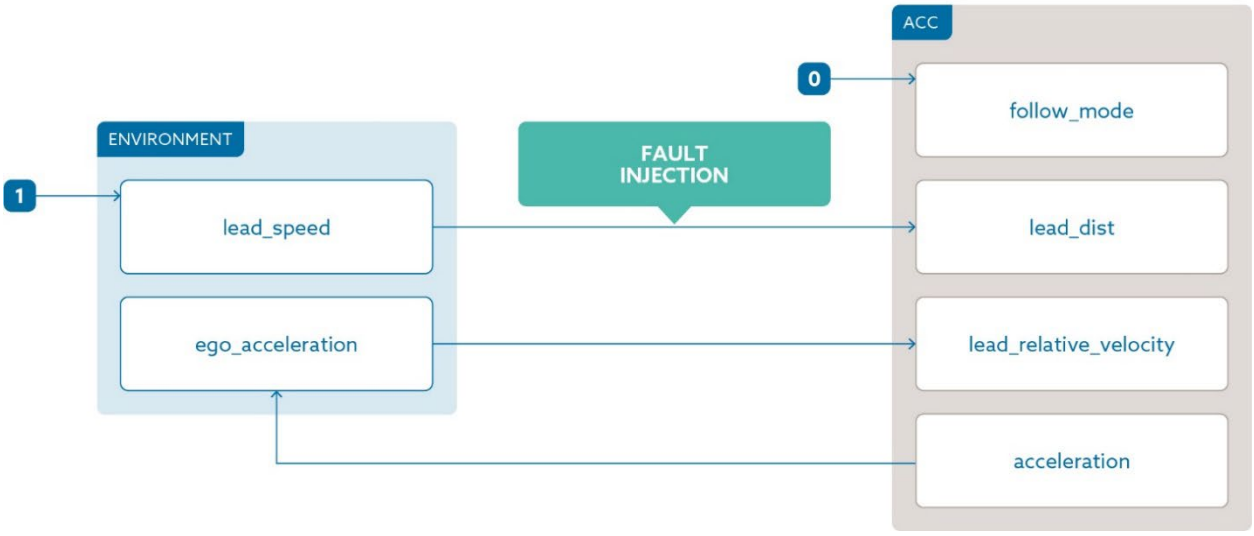
Generic Fault Injection MIL Testing Process



Example: Adaptive Cruise Control (ACC)

Consider an adaptive cruise control system that is tested against an environment model which provides the ACC model under test with its required inputs, including a simulated sensor value for the distance to the preceding vehicle.

Exemplary Simplified ACC Model



The fault tolerance specification states that a (sudden) failure of the distance sensor must not cause the ACC to trigger a hard (potentially hazardous) deceleration.

In this example, the fault is injected to the stimulation of the model under test inside the closed loop (type b. of the above enumeration). Based on the electric characteristics of the sensor, the test engineer identifies two different logical signal values that would likely be fed to the system upon sensor failure: 0 and NaN (not a number).

The engineer provides a parameterized test specification that accelerates the lead car to 40 km/h within 8 seconds and remains constant thereafter. The ACC is enabled. Thus, the ego car follows the lead car at a distance of 40 m. Two seconds after the final speed of 40 km/h has been reached, the fault is injected by instantly overriding the distance value by the parameter. The simulation is continued for another 5 seconds. During the entire time, at least the following signals were captured: lead car speed, ego car speed, lead car distance, and acceleration. Note that some of these are internal signals of the environment.
Two test case implementations are generated with the respective values of 0 and NaN for the parameter and automatically executed. For the evaluation of the test, the acceleration output of the ACC model is scanned for values below the legal threshold. After that, a fault-injection test report can be generated and, depending on the findings, passed back to the model engineers or to the archive.

Requirements concerning Test Specifications
The table below lists some requirements and characteristics of the presented example to allow a quick comparison to other user journeys based on common criteria.

Requirement	Evaluation
Requires scenario	No
Requires hardware	Simulation platform (usually a PC, server, or computing cluster)
Requires coordination of tools and models	Yes: Test-harness creation and feedback to evaluation algorithms (usually both automated)
Requires exchangeability among test instances	Not required but beneficial
Configuration of stubs/drivers/mock-up	In large parts automatable/derivable from the model
Kind of interface between test and scenario	Scenarios may include fault definitions that can be used as basis for fault-injection MIL testing

5.2.4 Scenario-based Open Road Testing

This user journey describes open road testing as it is used for the validation of ADAS functionality. The term open road refers to the fact that the system under test (SUT) is assessed in an open environment on public roads. This makes it on the one hand very difficult to control all test parameters in a desired state (especially the environment and the behavior of other vehicles), but on the other hand allows the SUT to be tested under diverse and unknown environmental conditions.

Motivation
Open road testing is used at the end of the V-cycle of a component with ADAS functionality. Beforehand, the basic functionality of the SUT is ensured by using other test methods like proving-ground testing that can control all test parameters and that can reproduce identical test conditions if necessary. Open road testing on the other hand can test the SUT with a broad coverage of different and complex environmental situations. Moreover, it allows the tester to assess the vehicle under test (VUT) in the real world with all its unforeseeable circumstances, and in this way gives a very good idea how the SUT would behave when used in production vehicles. Open road testing gives feedback on functionality in unknown scenarios where later analysis can reduce the number of unknown/unsafe scenarios for the SUT.

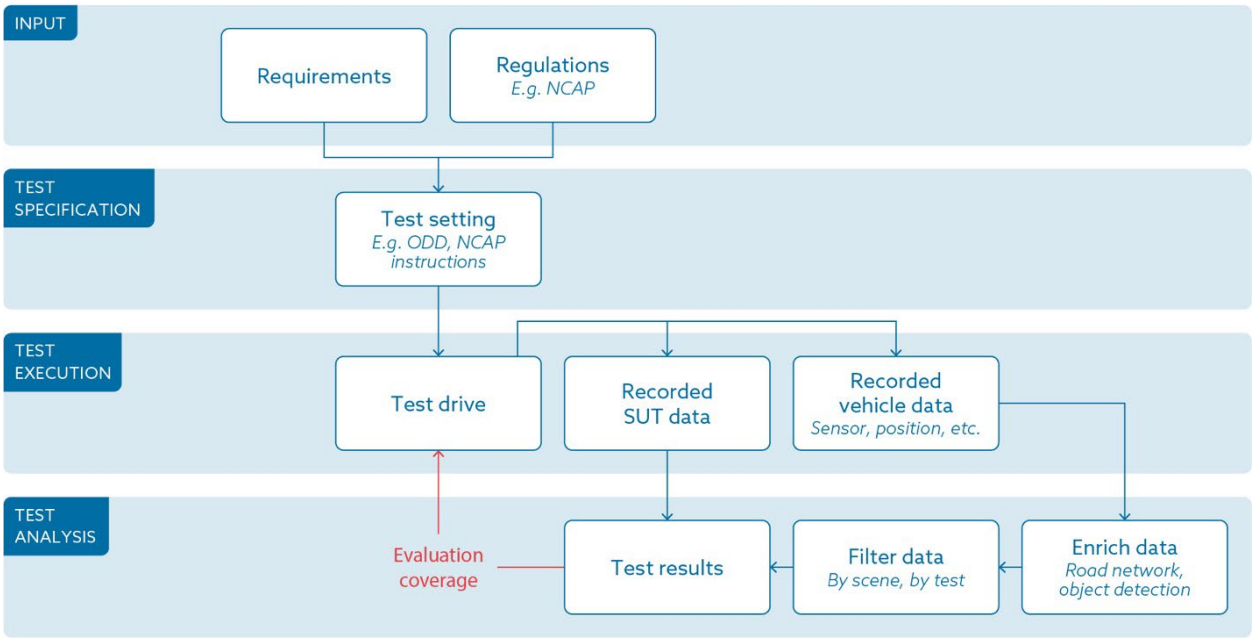
User Journey
At the beginning the user must define tests by writing test specifications. Open road tests can be divided into four different categories:

- Completely free test drives with little or no specification to examine the function of the SUT on the open road
- Test drives with unspecific and semiformal driving conditions using an operational design domain (ODD) or driving instructions from regulations like NCAP. The specification may include only the general setting (urban/rural/motorway) or an abstract definition of junction situations.
- Abstract test specifications including an ODD or abstract scenario descriptions (comparable to functional and logical scenarios from the PEGASUS definition)
- Concrete test specifications including concrete scenarios. This type is difficult to realize due to the uncontrollable environmental conditions in open road testing

In the latter two categories a specific route and time for the test drive can be selected to include specific junction scenarios and to determine weather and traffic conditions. In addition to the general setting, the test specifications define the variables and parameters of interest which must be recorded during the test drive.
It is important to emphasize that in most cases it is sufficient to only define the operational design domain of the tests instead of using scenarios. This allows a more general view on the SUT and on

the different situations that can occur during the execution of the test. ASAM will soon (November 2021) release an ASAM OpenODD concept paper Regulations like the NCAP regulation for automated lane keeping systems (ALKS) often play a big role when designing open road tests. In this case, instead of an ODD or a scenario description, the regulation itself will provide driving instructions and thereby define the open road test.
After the specification phase the SUT is assessed in the predefined setting, where typical durations of test drives are around an hour. The main limit for this time is a high generation of data, which at some point exceeds the large capacity of the on-board recorder. Dependent on the use case more drives are planned and executed to generate enough data for coverage. In addition to all the vehicle data, the behavior of the SUT is monitored and all associated data is recorded for later analysis.

Open road testing



Usually, the recorded data is not preselected during the drive but all data is recorded on a SSD hard disk with large capacity (around 10 TB). This data contains raw data from sensors (camera, lidar, and radar if needed), which takes up most of the space, as well as vehicle data such as speed, acceleration, and steering angle. Additionally, localization data, e.g. from GPS devices, is recorded. One possibility to reduce the amount of recorded data is to restrict recording to relevant situations. This approach is investigated by the KIsSME research project, which is described in chapter 4.3.3.4.
After the test drive the raw data is processed and enriched. As an example, objects are identified from raw sensor data and tracked (if not already done on board) and the associated road network description is created and linked to the localization data of the vehicle.
As a next step the data is preselected in different ways:

- Focus on relevant situations: Relevant situations that assess the functionality of the SUT are identified. Data associated with these situations are selected by time
- Focus on test specifications: Situations that satisfy conditions of previously defined tests are selected by time
- Focus on data type: Data which are needed for the evaluation of a specific test or situation are selected by type (e.g. just raw data from a specific sensor if this sensor is to be examined)

These steps will result in final data which exactly defines concrete test specifications. By analyzing the VUT data recorded during the test drive, the associated test results are generated. Together they result in concrete test cases which give valuable feedback on the functionality of the VUT during the open road test drive.
The final test data might be used to generate standardized scenario files which then can be used as an input for simulation-based testing.
Dependent on the outcome of the analysis, more test drives are planned with adjusted settings. If for example the VUT performed especially badly in specific weather conditions like rain combined with a specific traffic situation like a traffic jam, more test drives with a high probability of this combination will be planned and executed. The variability of open road testing allows a flexible readjustment of the testing in process together with the advantage of having very realistic test conditions.

Requirements concerning Test Specifications

Requirement	Evaluation
Requires scenario	optional, depending on the use case
Requires hardware	Test vehicle
Requires coordination of tools and models	Yes: Mostly analysis of recorded data (enrichment and preselection/analysis of data)
Requires exchangeability among test instances	Not required but beneficial
Configuration of stubs/drivers/mock-up	Similar to production vehicle since at the end of the V-cycle. Test driver for operation is required
Kind of interface between test and scenario	Usage of operational design domain or optionally abstract scenarios at the beginning to define the setting of the test. Possible usage of concrete scenarios or ODD descriptions as test artifacts or as generated result of the test drive
Covered by best practice and regulations and standards	ISO OTX (test sequences), ASAM MDF (recorded data), ASAM OpenODD, optionally ASAM OpenDrive (road networks) and ASAM OpenScenario for part of the scenario descriptions

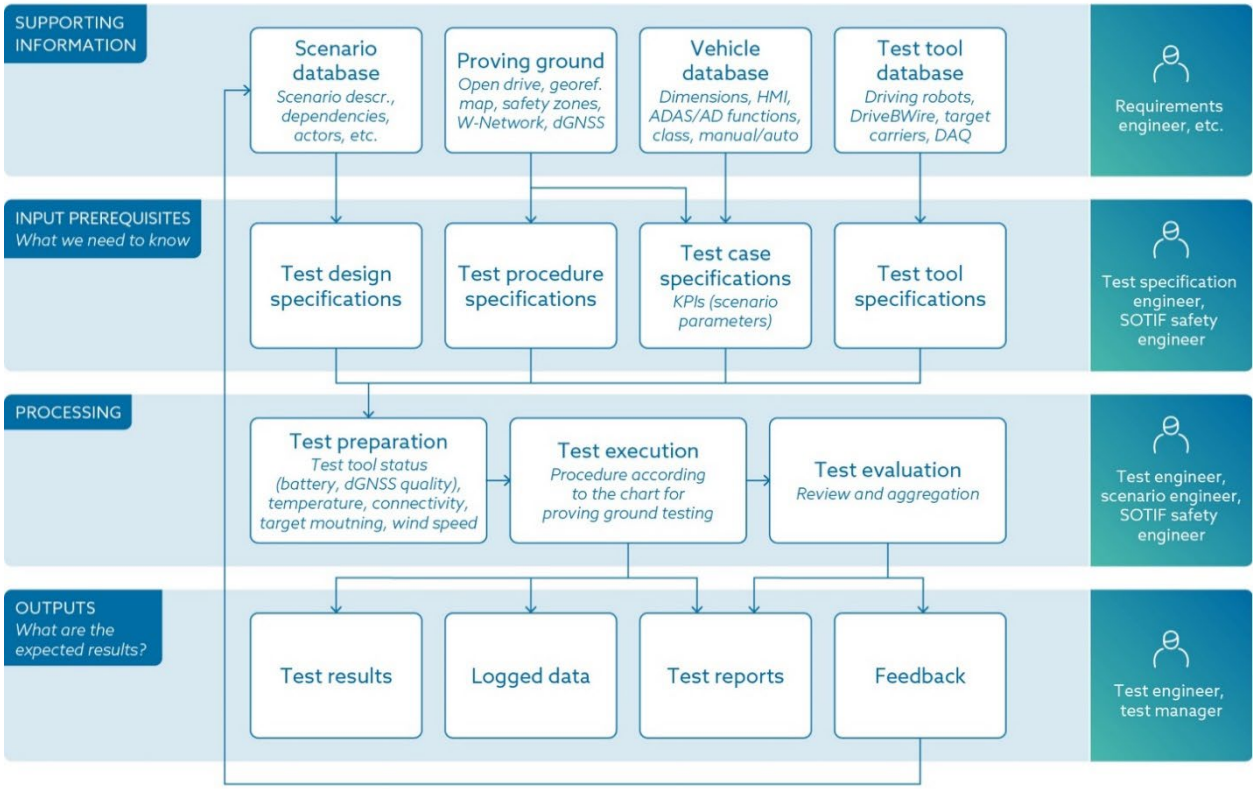
5.2.5 Scenario-based Testing on Proving Grounds

This section presents scenario-based testing of highly automated vehicle functions in the area of the proving ground. The following user journey demonstrates the high-level view of the workflow. A possible cut-in scenario for the traffic jam chauffeur use case is described in the procedure. The focus of the examples is on the interaction between test cases, scenarios, and test-environment-specific conditions.

User Journey

In this user journey, the different phases for the validation of ADAS/AD functions in the area of scenario-based testing are shown. Here, the focus is on the testing of highly automated vehicle functions in the area of the proving ground.

Proving Ground Scenario-Based Testing



Tools involved, interactions, and interfacing required:
driving robots, DbW interfaces, target carriers and targets, proving ground, dGNSS base and positioning systems, software control center, etc.

Derive Test Case Descriptions from ASAM OpenSCENARIO

Mapping a simulated scenario out of an OpenSCENARIO description requires additional information to cover all the needs of the proving ground. The ISO 22133 metalanguage describes

the behavior of individual participants within a scenario on the proving ground. Every participant needs to have an individual scenario description based on this metalanguage. One possible approach is to translate OpenX information into this language. Using the ISO 22133 description language, a scenario is described by a text file that consists of several lines. Each line is defined by a given type:

Object

- Described by an ID and its physical dimensions

Event

- Contains information about a condition whose fulfilment is to be checked
- Phase contains information about a trajectory segment and a list of events which can cause phase transitions during the execution of the trajectory segment. The parametrized OpenDRIVE coordinate is used as the desired position for the given object.

This information must be transformed to the ISO 22133 protocol messages, which are controlling the individual participants.

The device interface description (DIDX) describes the vehicles' supported ISO 22133 messages and additional information, limitations, and parameters of the test object. The DIDX file is XML based and split into seven main sections. It must implement the DIDX schema as specified in this document.

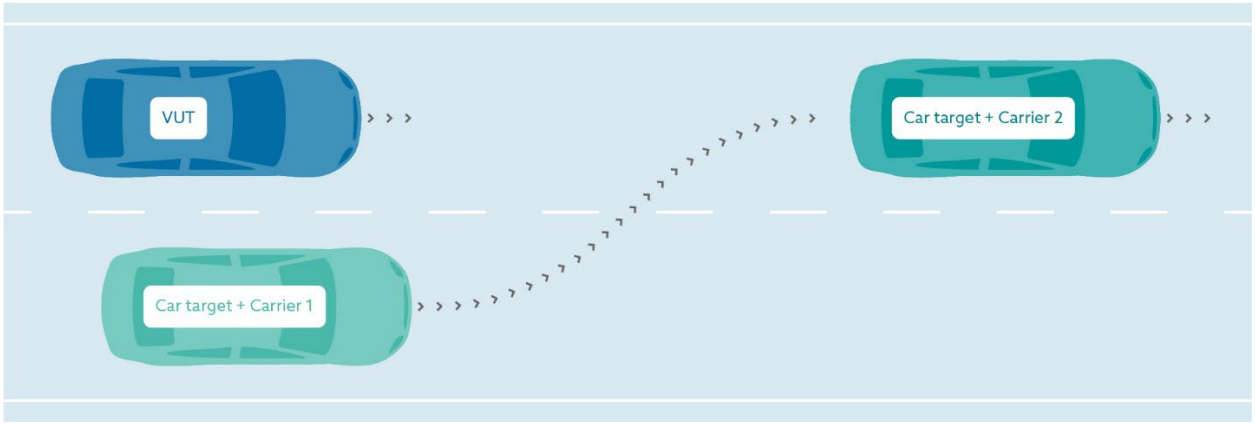
- Information: This section contains some general information about the file and the test object. All information specified in this document must be included in the DIDX. Vendor-specific information may be added
- Limitations: This section contains the limitations of the test object. The limitations and the corresponding units are specified
- Messages: For each message the test object supports, an entry must be made in the DIDX
- Data types: Special data types are specified in this section and can be referenced by another data type or the content of a message. A data type can be a structure, an enumeration, or a bitfield
- Parameters: This section describes the parameters of the test object, which can be accessed via the control center. Each parameter has a name, a parameter ID, an access type, and the data type
- Error codes: This section defines the error codes from the MONR message
- Protocol tunnel: Defines the ID, name, vendor, and the version of the protocol which is tunneled through the ISO protocol

Requirements concerning Test Specifications

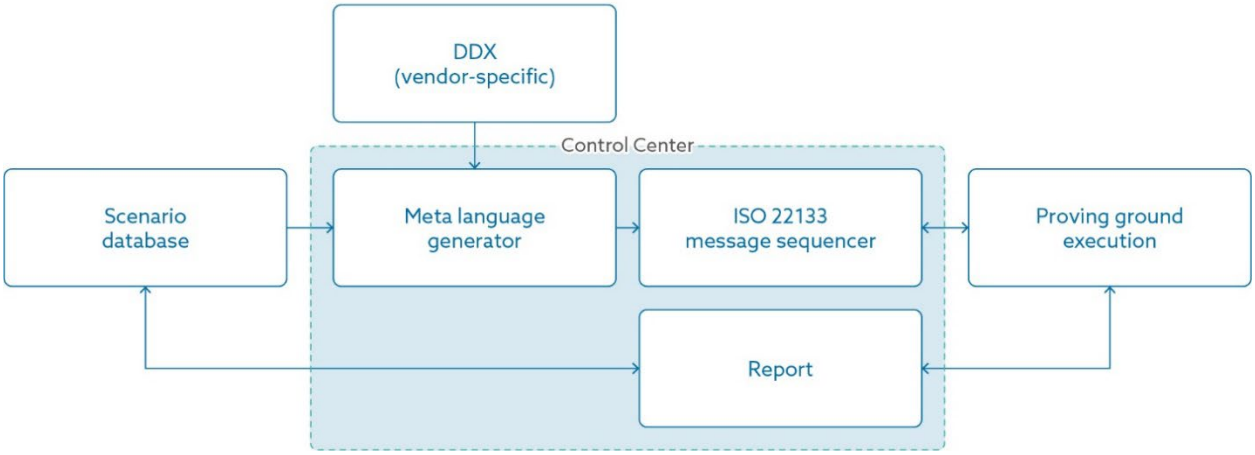
Requirement	Evaluation
Requires scenario	Describes the behavior and trajectories of each participating actor within the scenario. Actors

	can be moveable but also stationary test objects
Requires hardware	Specification of actors like vehicles, robotic platforms, steering robots, measurement systems, periphery devices (e.g. weather stations, traffic lights, lane markings)
Requires coordination of tools and models	Describes the behavior and trajectories of each participating actor within the scenario. Actors can be moveable but also stationary test objects
Configuration of stubs/drivers/mock-up	Parametrization of actors and measurement systems
Kind of interface between test and scenario	Description of data interfaces between actors
Covered by best practice and regulation standards	List of all standards to be complied with

Cut-in Scenario Example, Traffic Jam Chauffeur Use Case



Proving Ground Testing Procedure



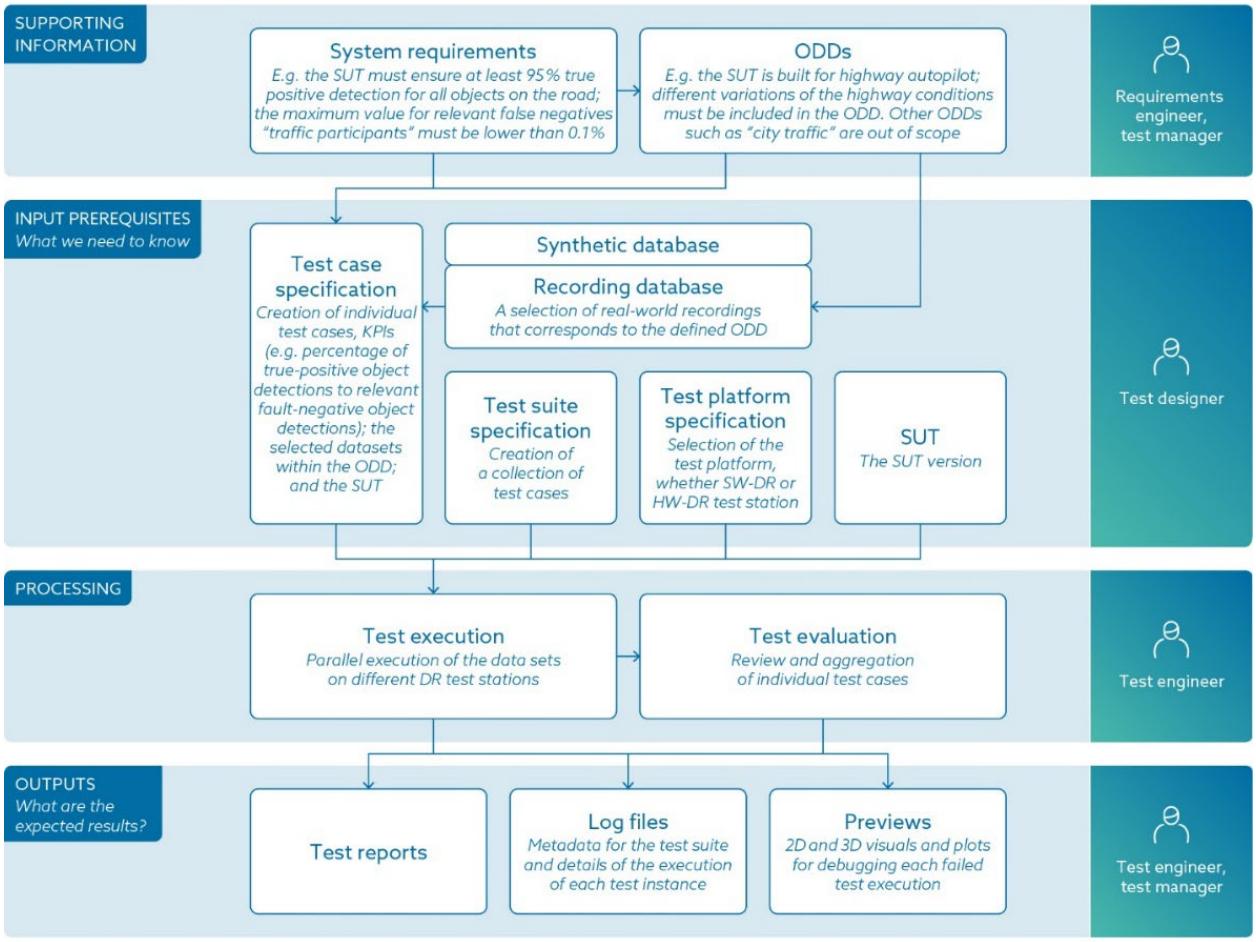
The task of the control center is to control all test objects on the proving ground. To be vendor independent, the OpenSCENARIO standard is used to describe the test scenario. It is mandatory for the control center to be able to parse OpenSCENARIO and OpenDRIVE files as input.
To control test objects such as driving robots and target carriers, ISO 22133 is used. This provides the possibility for using test objects and control centers from different vendors.
In order to perform a lane change the open scenario file needs to be provided. The control center internally parses the file and translates it to ISO 22133 messages to upload the scenario to the test objects. Due to the different dynamics of the test objects, the DIDX is used to pre-check all limits and may make some changes to the scenario in order to perform the test. All test objects send their telemetry data over the network to operate control loops if necessary. Telemetry data is recorded by the control center for postprocessing and evaluation.
A control center may give an initial report if the test was performed correctly.
Conclusion:
The suitability of OpenSCENARIO for the Proving Ground use case seems promising but needs further investigation.
ISO 22133, Road Vehicles – Test Object Monitoring and Control for Active Safety and Automated/Autonomous Vehicle Testing, specifies requirements, functionality, and a protocol to control manufacturer-independent target carrier systems. To provide all the information that is missing in OpenSCENARIO, ISO 22133 contains a device interface description (DIDX) and a scenario description metalanguage.

5.2.6 Hardware Reprocessing / Data Replay

This section introduces data replay (DR) testing. DR is an open-loop test methodology that is based on replaying recorded data sets to the interfaces of a system under test (SUT) and evaluating the responses of this SUT against reference or ground truth (GT) data. Such a test methodology can be deployed on multiple test platforms, whether as software data replay (SW-DR) or as hardware data replay (HW-DR).

User Journey
In the following figure, the user journey for the validation of an environment perception component with data replay is presented. The multiple phases of the test journey can be generalized to cover not only the example of environment perception components but the whole ADAS/AD function spectrum. Depending on the nature of the SUT, the test platform could take the shape of a purely software test platform (SW-DR) or a hardware test platform (HW-DR), once the SUT is deployed on the target system on chip (SoC).:

Testing of an Environment Perception Component with Data Replay (SW-DR/HW-DR) Verifying That the SUT Performs Correctly within the Defined ODDs under Different Traffic Scenarios



Tools involved: data replay test platform (SW-DR or HW-DR test system), including SUT interfaces, e.g. bus, sensor interfaces; data management software, including recorded data sets; and test management software, including the abstraction of individual tests into high-level representation, e.g. test suites

The whole test starts with the requirements definition. In this phase the requirements engineer and the test manager define the scope of the DR test campaign. Once the SUT success and failure criteria are identified, such as the given example here with the percentage of positive object detections, the operational design domain (ODD) of the function must be clearly identified. This is an important step to utilize the data management software to select the relevant data sets within this ODD out of the petabytes of recorded data.

The test designer then takes this specific task of selecting the right data sets out of the data lake. In addition, he/she creates the test cases for each category of these data sets, selects the DR test platform (SW-DR or HW-DR test station), and selects the SUT version (always the latest SUT version or a fixed released version). He/she is also responsible for the grouping of the multiple test cases within a test suite so that the results of the individual test executions are aggregated in a single viewpoint.

Once all the test specifications have been performed, the test engineer takes over the automatic test execution. The tests can be executed iteratively or parallelly. Parallel execution is preferred in order to shorten the test cycle and correspondingly the time to market. The results are then shown in a test report with all test result artifacts attached to it, such as the execution log data, including the test metadata. Moreover, a preview tool enables test debugging if a test failure exists.

Requirements concerning Test Specifications
The table below lists some requirements and characteristics of the presented example to allow a quick comparison to other user journeys based on common criteria.

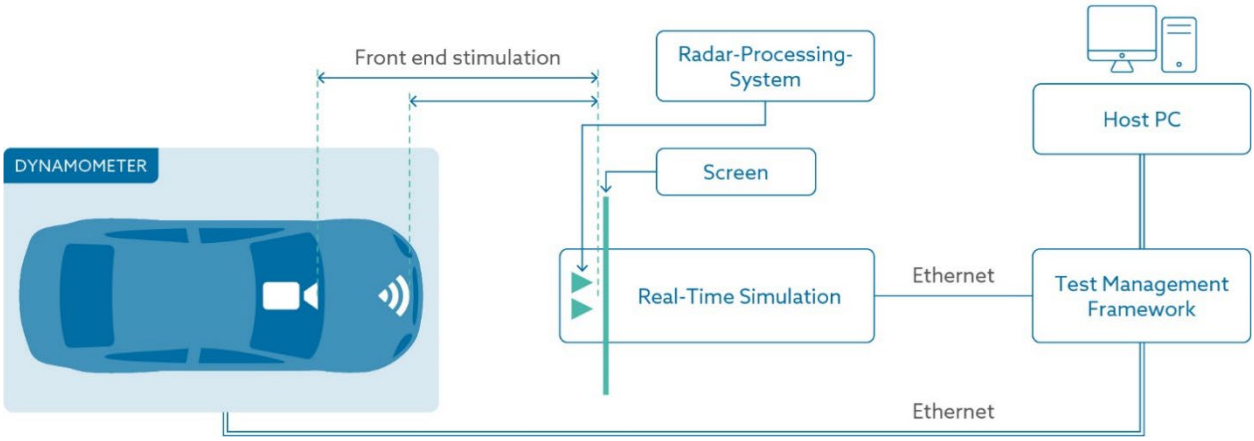
Requirement	Evaluation
Requires scenario	No. A scenario is not required. A selection of recorded data sets out of the data lake, according to the criteria of the ODD, is used.
Requires hardware	According to the test target, SW-DR and HW-DR test stations could be selected. For functional testing, especially at the beginning of the software development, SW-DR is preferable for the ease of scalability and parallelism. HW-DR is preferred in later development cycles for function testing, as well as for robustness testing in different traffic scenarios. Both test platforms are usable for failure insertion on bus/network as well as sensor data streams
Requires coordination of tools and models	Coordination of data replay test platform, replay data management, and test management
Requires exchangeability among test instances	A single SUT can be tested with multiple data sets, making the exchangeability of test artifacts across different test instances relevant
Configuration of stubs/drivers/mock-up	Optional
Kind of interface between test and scenario	Does not apply

5.2.7 Requirements-based Vehicle-in-the-Loop Testing

This section presents the functional testing of autonomous driving/ADAS functions using vehicle-in-the-loop (VIL) testing. The user journey shows the different steps which are typically required to execute such a test on largely unmodified vehicles.

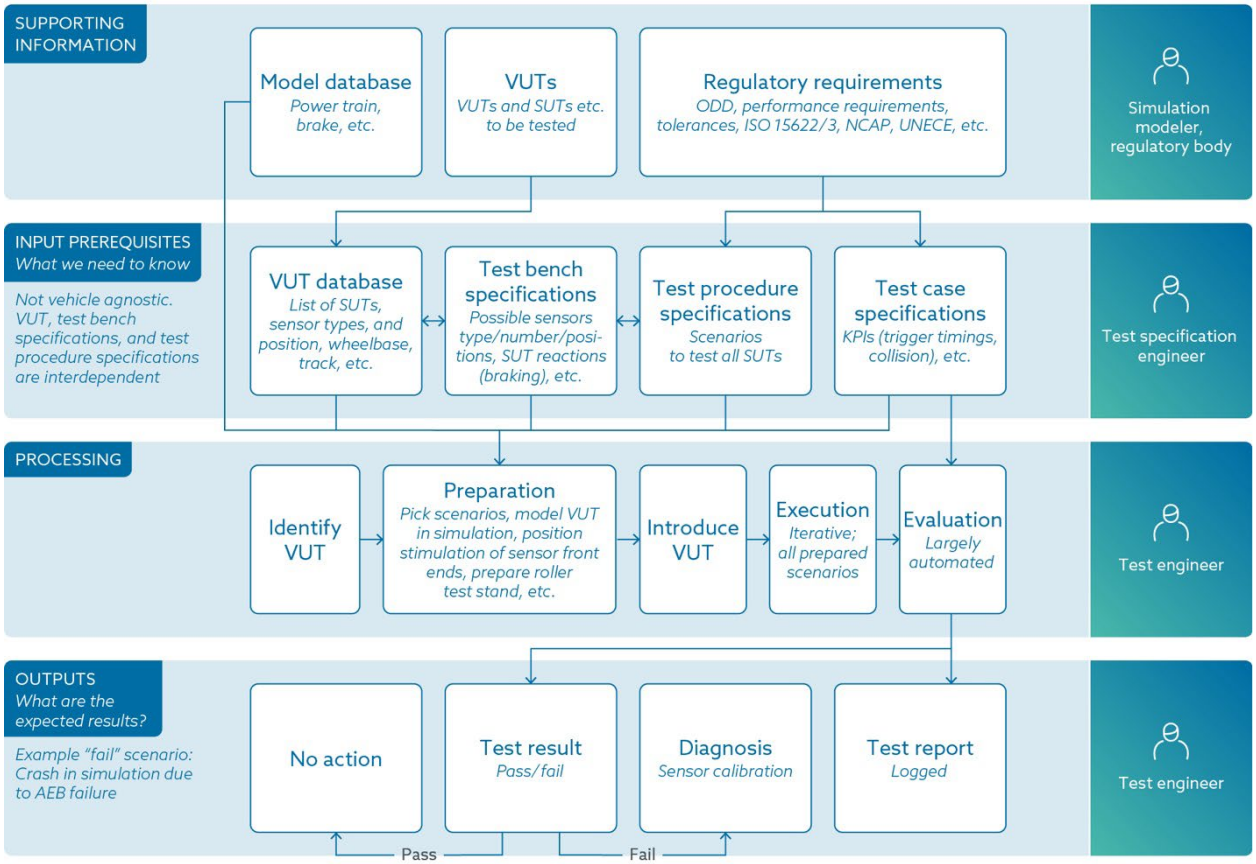
User Journey
The framework described here to apply VIL testing of ADAS functions to vehicles which do not have to be specially modified to enable testing is described in this user journey. This is possible by employing exclusively over-the-air (OTA) technology to stimulate the sensors instead of using data injection. This enables the evaluation of the full sensor set and the complete chain of effects (this includes the physics of the sensor and the full software stack). With this it is possible to test the full system reaction.

Test Setup VIL



This potentially also enables the application of functional VIL testing of ADAS functions to periodical technical inspections (PTI) and production testing. VIL test benches which rely on heavily modified vehicles can only be applied in research and development.

VIL Testing of ADAS Functions with Over-the-Air Sensor Stimulation of Sensor Front Ends with Target Simulators of Front Camera and Radar. Test Idea: Trigger Functions of Largely Unmodified Vehicles on Roller Test Stand. SUT: LKAS, LDWS, FCWS, AEB, ACC



Hardware: Real-time simulator, roller test stand, OTA camera simulation, OTA radar simulation, OBD interface
Software: Scenario simulation, hardware interfacing and control, database interfacing, test automation, automatic report generation, user interface

In a first step the driving functions to be tested must be defined and the requirements must be formulated by the simulation modeler or by regulations. The test specification engineer then derives test specifications and scenarios to satisfy the requirements.

Based on these and the chosen vehicles under test (VUTs), a test bench consisting of multiple hardware and software components (see "Technical Application and Tools Needed") is assembled and a continuously maintained database of VUT and test spec details is set up.

Before test execution the database is used to precondition the test bench for the specific VUT. The test is then executed by a test engineer fully automatically or semi-manually, depending on the driving function. The test results are then evaluated, matched to the test case specifications, and a test report is generated. Depending on the area of application this test report may be more detailed or simply a pass/fail that leads to further diagnosis.

5.2.8 Scenario-based SIL Closed Loop Testing

This section presents an example of a validation approach for a camera-based ALKS (automated lane keeping system) as a V-ECU (virtual electronic control unit, **https://www.prostep.org/fileadmin/downloads/WhitePaper_V-ECU_2020_05_04-EN.pdf**) on vehicle level by means of a scenario-based test. It is purely simulation based and focuses on the technical aspects of SOTIF (safety of the intended function, ISO DPAS 21448) for a positive risk balance for unknown risks. The presented user journey is intended as an example only; the presented workflow, terms, and roles have no guiding character. Subsequently, examples for the emergency brake assist (AEB) and the automated lane keeping system (ALKS) illustrate this user journey in detail.

User Journey
The basic test idea is the comparison of an ALKS system and human driver capabilities in a closed-loop Software-in-the-Loop test bench. The SUT (system under test) is a camera-based ALKS closed-loop system as software stack in the form of a V-ECU. The driver, the vehicle, and the driving environment are virtual and are implemented as models and as scenario and map data.
The test goal is to find unknown risks by exploring and generating scenarios and finally provide evidence of a positive risk balance for the ALKS compared to a human driver for these scenarios.

Supporting Information
As supporting information and data the system requirements, the ODD (operational design domain), a scenario database, credible simulation models, and qualified tools (e.g. simulator) are required to define and implement the test specification. In the example the system requirements say that the focus is on the evidence that ALKS does not introduce unreasonable risks compared to a competent human driver. The ODD is for speeds below 60 kph and expects no pedestrians on the road (e.g. motorway). It aims at the usage of ALKS for passenger vehicles only.

Input Prerequisites
All the supporting information is turned into a complete and consistent test specification with this content:

- The test design specification is a “document specifying the features to be tested and their corresponding test conditions” [ISO 29119]
- The test case specification is a “documentation of a set of one or more test cases” [ISO 29119]
    - It specifies (concrete) test cases consisting of concrete scenarios and one or more evaluation criteria
        - E.g. use TTC (time to collision) and collision events as metrics including expected results (e.g. TTC never below 1.0 sec).
        - Vary scenario/map parameters (e.g. road curvature)
        - Create new scenarios by reordering existing scenarios

- The test procedure specification is a “document specifying one or more test procedures, which are collections of test cases to be executed for a particular objective [in execution order]” [ISO 29119]. Additionally, the test procedure specification may also specify zero or more test bench configurations that shall be used to execute the test case(s).
    - Test bench configuration: e.g. SIL environment, scalable (cloud), qualified tools
- Definition of the SUT (or also called test object): E.g. ALKS as V-ECU Level 1, Version 8.3

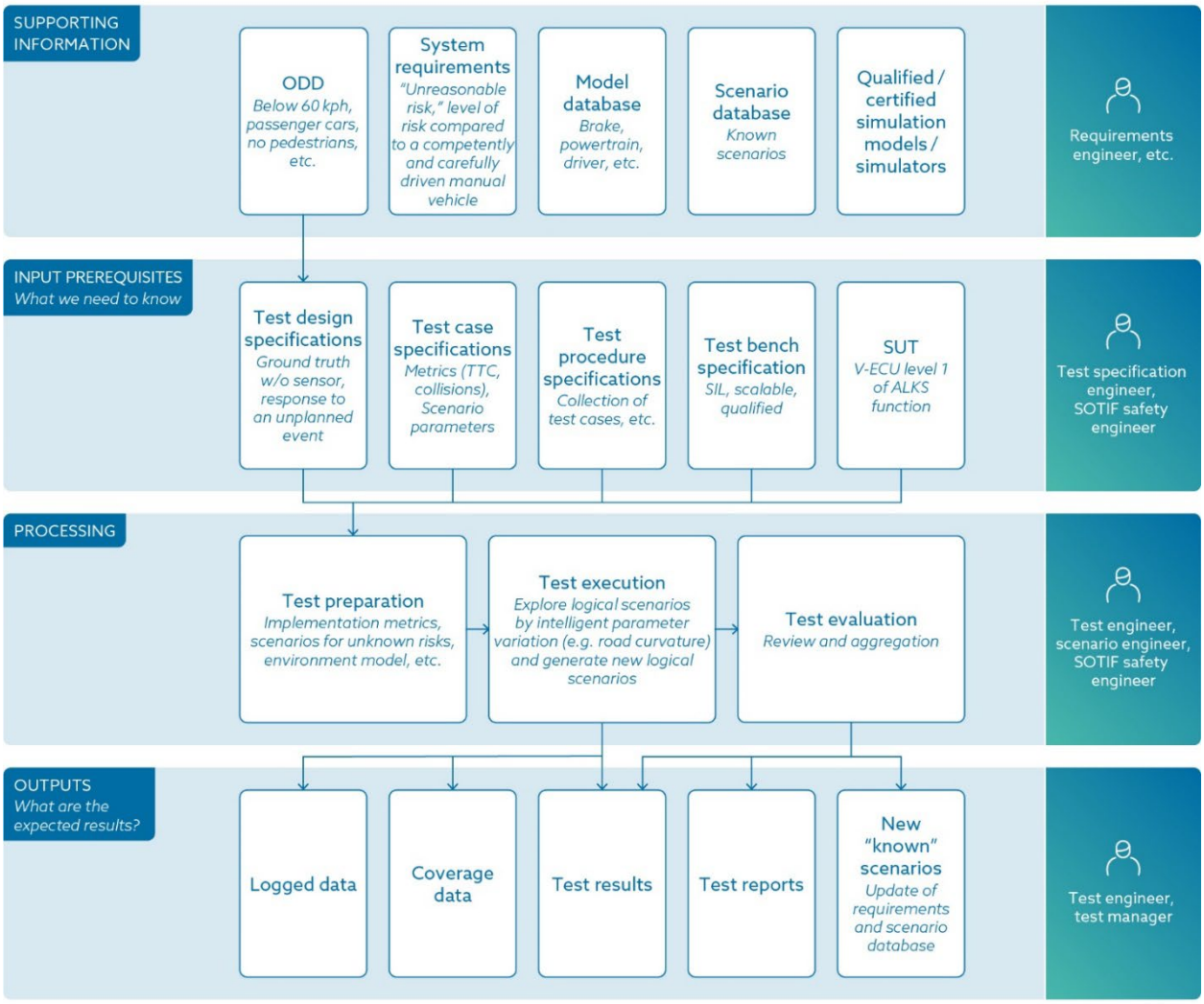
Processing

The processing phase is divided up into the steps test preparation, test execution, and test evaluation. During the preparation phase all the specified metrics, models, and scenarios are put in place according to the test specification given. During the test execution many simulation runs are performed, each on a new concrete scenario. The concrete scenarios were created by parameter variation (e.g. road curvature) of logical scenarios/maps.
During the test evaluation specified metrics are applied to the simulation results to produce the test results and to generate aggregated information from all the test results.

Outputs

Test execution and test evaluation produce test results and further data for analysis. In this way unintended ALKS behavior is detected, can be analyzed, and finally be improved by the ALKS development team. New scenarios which turned out to be important but have been “unknown” prior to this testing are an additional result. They are fed into the scenario database and are deployed in the subsequent testing activities to continuously measure the positive risk balance (KPI).

Validation of Camera-based ALKS System as V-ECU on Vehicle Level by Means of a Scenario-based Test. Technical Aspects of SOTIF: Positive Risk Balance for Unknown Risks. Test Idea: Comparison of ALKS and Human Driver in a Closed-Loop SIL Test Bench. SUT: Camera-based ALKS closed-Loop System as Software Stack



Tools involved, interactions and interfacing required:
test software, SIL simulator, test management software, etc.

Requirements concerning Test Specifications

The table below lists some requirements and characteristics of the presented example to allow a quick comparison to other user journeys based on common criteria.

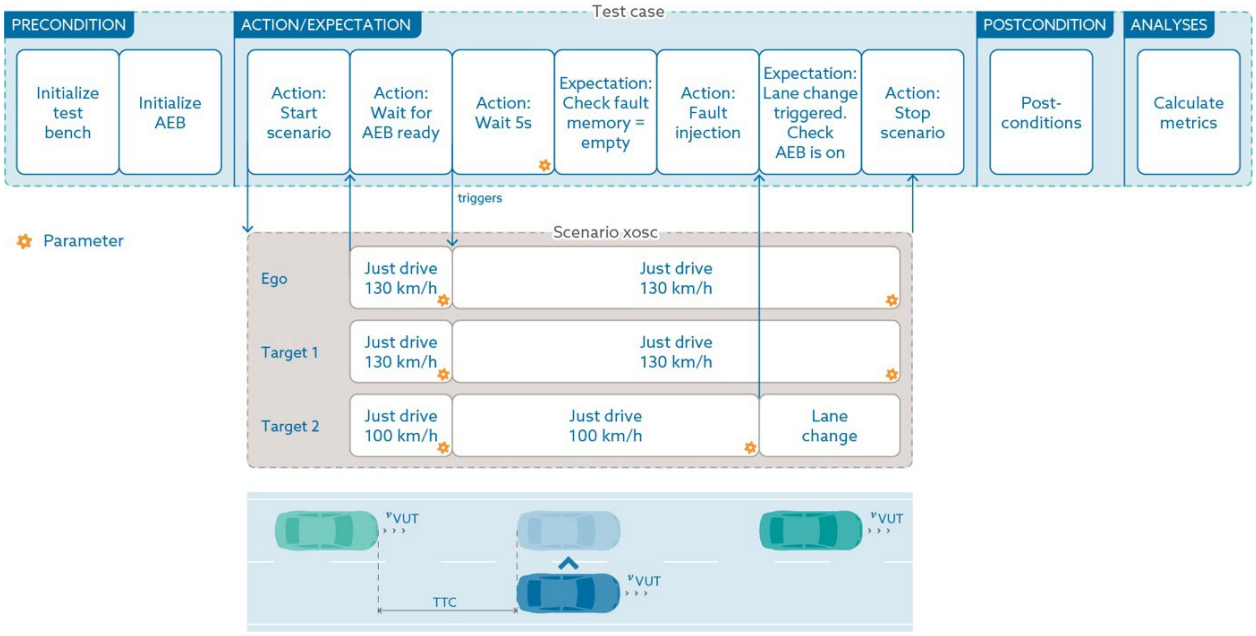
Requirement	Evaluation
Requires scenario	Yes, basic technique for the presented example is to generate and explore scenarios
Requires hardware	No hardware required; pure simulation-based approach

Requires coordination of tools and models	Yes: Evaluation, exploration, scenario generation need to be coordinated: Test software, SIL simulator, test management software, scenario database
Requires exchangeability among test instances	At least the newly created scenarios go to the pool of “known scenarios” for other test activities
Configuration of stubs/drivers/mock-up	n.a.
Kind of interface between test and scenario	The test explores and generates new scenarios
Covered by best practice, regulations, and standards	Yes, approach is based on ISO/PAS 21448, safety of the intended functionality (SOTIF)

Examples

Three different examples for testing driving functions in different ways in scenario-based testing are described below. The focus is on the relationship between test cases and scenarios.

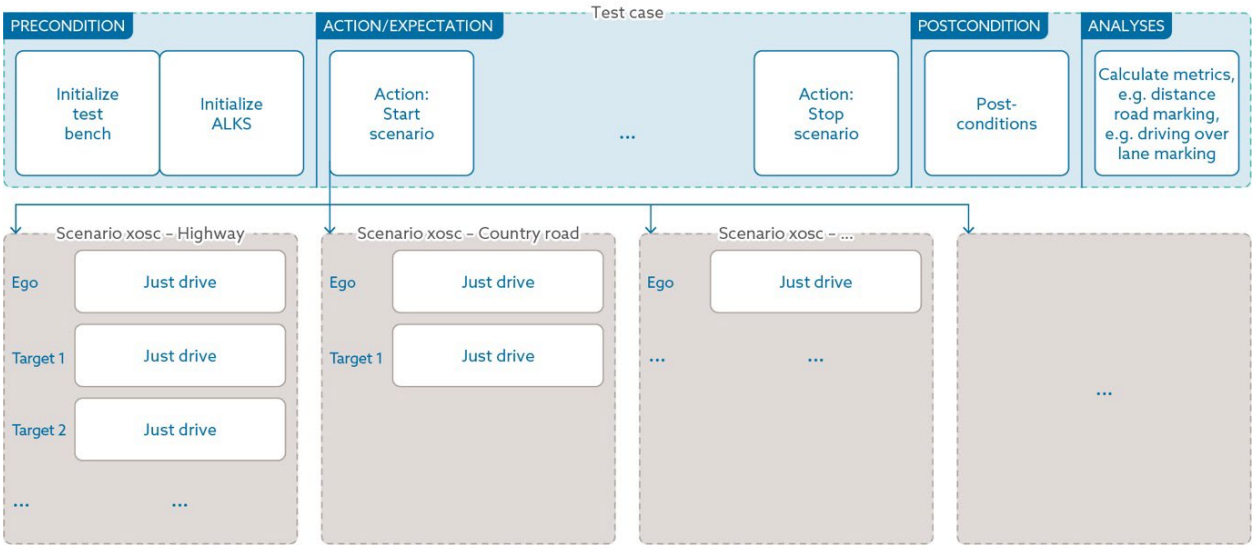
Example 1: Testing the Emergency Brake Assistant
Specifics: Test Case Reaction to Certain Maneuvers of the Scenario



Focuses of Example 1

- Strong coupling of test case and scenario
- Parameterization of logical scenarios and logical test cases as reusable, independent units
- Trigger from scenario to test case to execute action
- Trigger from test case to scenario to execute next maneuver steps

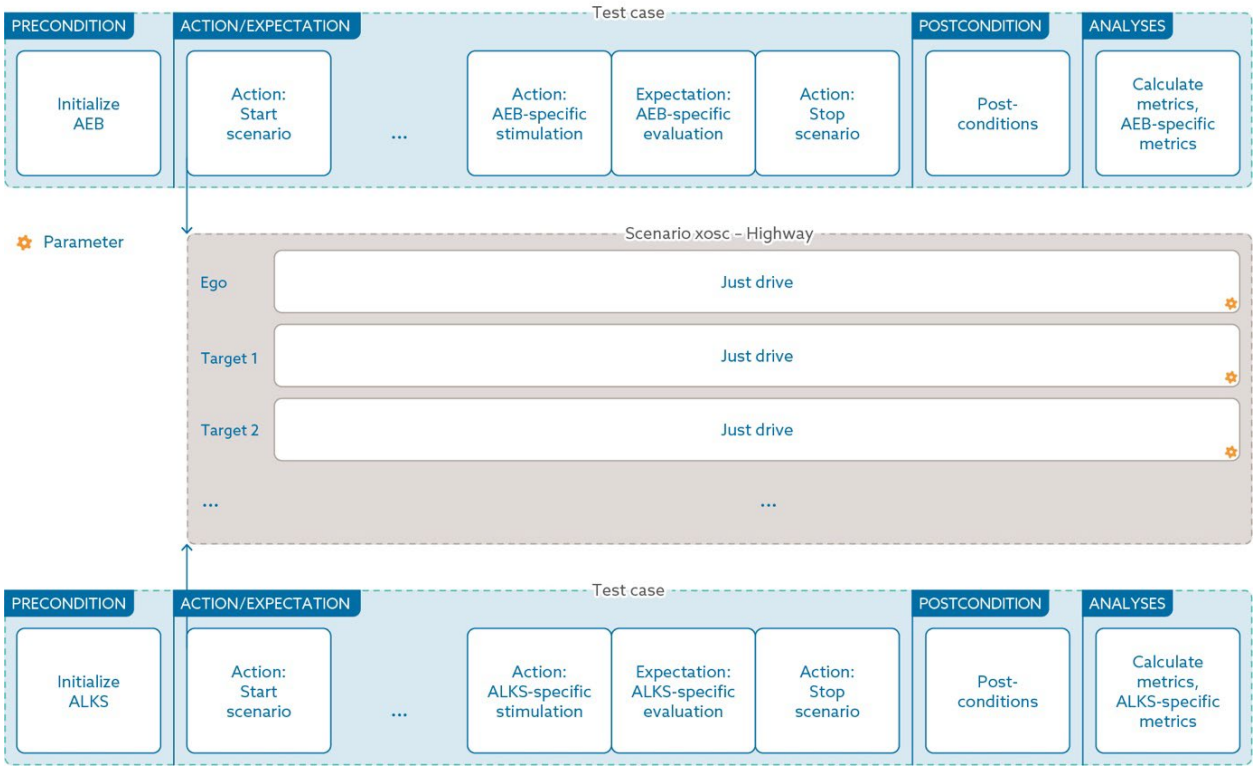
Example 2: Testing the ALKS for Different Scenarios
Specifics: Reuse Test Case for Different Scenarios



Focuses of Example 2:

- Reuse of artifacts
- One test case is executed with different scenarios
- One scenario is executed with different test cases

Example 3: Testing AEB, ALKS, etc.
Specifics: Reuse Scenarios for Different Test Cases



Conclusion

The analysis of the different examples provides the following findings:

1) In some cases, test case and scenarios are more or less coupled with each other. The concrete type of coupling is still partly tool-specific and proprietary and thus not transferable between tools. For better transferability and maintainability, the coupling could be standardized. Existing standards, such as OpenDrive, OpenScenario, XIL, and OSI, might be used and extended.

2) Test case and scenario are two independent artifacts. At the same time, parameters are varied in the test case and scenario. Parameter variation should therefore take place across the two artifacts. The parameter variation as currently defined in OpenScenario is thus not sufficient and should be reconsidered.

5.3 AI Specific Aspects

The algorithm paradox

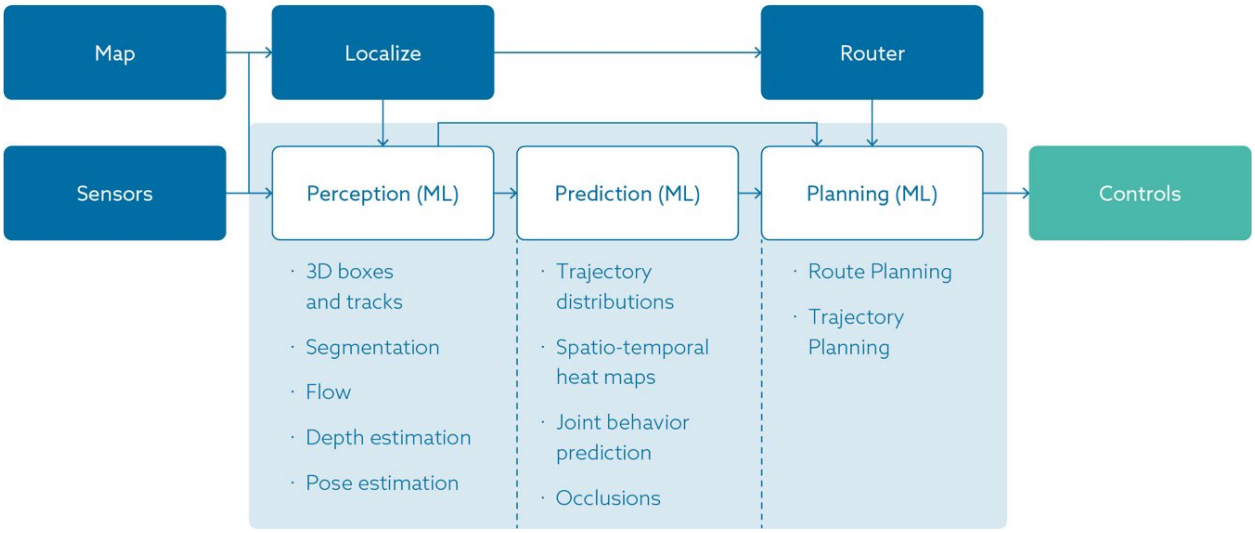
Although the boundaries of artificial intelligence have tended to shift over time, a core objective of AI research has been to automate or replicate intelligent behavior. Conditions encountered while driving are arbitrarily complex, and infinite-dimensional. As such, manually encapsulating and defining generalized rules that dictate safe and effective driving becomes impossible. By its ability to automatically learn complex rules from data, Machine Learning has emerged as the major paradigm to create ADAS/AD systems. For highly automated vehicles, especially on SAE Level 4 or 5, this means AI applications can enable the processing, selection or extraction and interpretation of data during tests in real-time, while at the same time monitoring itself.

Machine Learning (ML) Modules in ADAS/AD Software Stack

ML usually takes the form of *scene understanding* and *ego-vehicle planning* to various degrees.

- Scene understanding/perception: Understanding the world and recreating it in a model. This involves the two further steps of perception and behavior prediction
- Motion/trajectory planning: Navigating using the model as a proxy for the world

ML Applications in AD

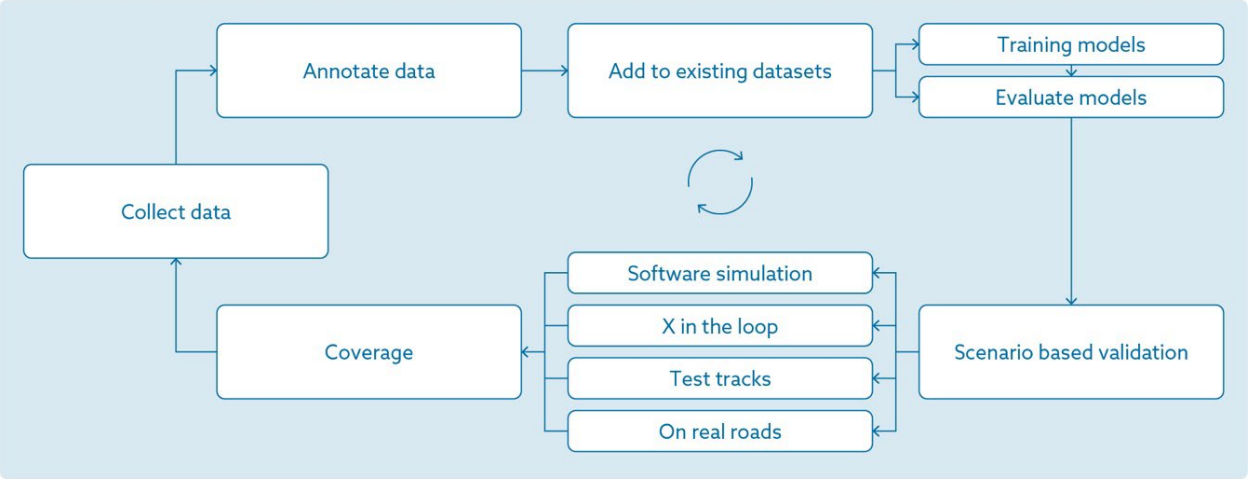


Source: Waymo

Development of ML Modules

As shown in figure, ML modules are created mainly using the steps in order of data collection labeling, data split training and evaluation.

Development of ML Modules

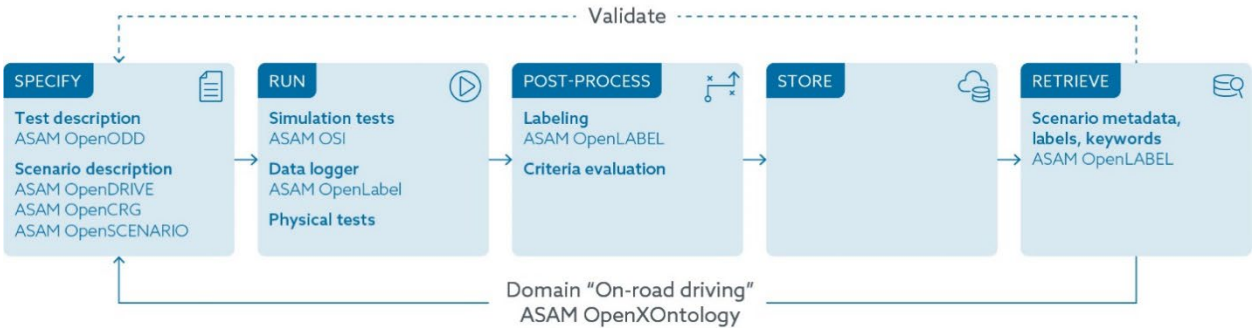


Impact of ML on Traditional ADAS/AD Testing Methodologies

ML development for ADAS/AD needs diverse driving data where ideally every scenario that might show up after deployment in the ODD of interest are present in a statistically significant way in the data set. However, the lack of a credible method to measure and leverage diversity coverage has necessitated blunt testing to ensure nothing is left out. In practice, it has started to manifest in two ways:

- On-road testing → high milage
- Simulation → Combinatorial combination of parameter values to create scenarios. The gold standard for this is scenario-based testing, which is duly expanded in the following figure

Scenario-Based Testing with ASAM OpenX



Suggestions for ML specific Testing Methodologies

While there has been significant progress in testing ML modules for ADAS/AV, they are all far from perfect. A few broad areas where considerable progress is still to be made are:

- Rigorous processes that *align ODD and intended functionalities* with data specification, data and ML architecture selection, training, evaluation, and monitoring

- Data specification: It has to align with ODD and intended functionalities. OpenLabel might be useful here. However, the challenge would be how OpenLabel can be made versatile enough for features of all kinds (visual, spatial relationships, temporal, and causality)

- Mapping data: Representativeness of the ODD to ensure sufficient coverage

- *Unmodeled concepts*: Potential solutions might include keeping OpenXOntology open including more objects. OpenLabel may support unknown objects, and containerize for example if any object does not fit the description

- *Data augmentation* by generation of synthetic data to fill in the gaps in the data diversity. This means artificially increasing the training data on what is missing – resize, rotate, brightness. Domain adaptation/randomization further helps in this regard as it is similar to concrete scenario generation for prediction, e.g. random vehicle textures (sometimes not even there in the data set). The whole process may lead to robustness which may be extended to all input modalities. The main difficulty is identifying the relevance of the variation dimension (e.g. contours might matter more than textures)

- Need for *realistic graphics* to validate perception-based ML modules in simulation. *Data redundancy* strategies such as sensor fusion and ensemble concepts. Redundancy always helps, such as in an airplane that has three systems working at any point

- *Coverage argument* for the ODD where data is used for both the *training and evaluation* of ML modules

ML Can Also Be Used to Make Traditional Testing Better

- Modern techniques like OpenScenario allow mapping an infinite-dimensional domain (highways for example) into millions of concrete scenarios by allowing easy definition of abstract scenarios with constraints. This concrete scenario generation is too tedious to evaluate and many of them are irrelevant in the ODD of interest

- ML can help take clues from driving data and help trim down the millions of scenarios to a few ones that are relevant to a given ODD

Challenges to Increasing Trust in AI

- AVs are safety-critical and hence it has to be provably safe, which increases the trust of all stakeholders including government and the public. Hence, the AI-based systems must be explainable, so that their behavior can be understood and assessed.

- They have to be robust against adversarial attacks and against perturbations in the input data that could potentially be caused naturally, such as by slight snow or soiling on the sensor.

- Behavior should be in bounds to the specification of the intended functionalities.

- Perceived safety is important because no matter how safe certain vehicle behaviors are, things like sudden jerks due to braking or acceleration or aggressive overtaking have the potential to scare the users.

-

V&V strategies for ML specific aspects must be defined and must improve the trust into AI.

6 Test Data Management

How to Model and Apply Test Data for the Future of Autonomous Driving

The previous chapters looked at the challenges and opportunities of new test strategies in detail, and the role data generation plays in it. By now, the relevance of test data management should be clear. Test data does not appear out of thin air. It must be planned and designed, modeled, and stored, before it can be used. Simply put, the more advanced the test data management strategy, the more efficient the testing of ADAS/AD functions becomes. If defects can be detected and identified early on in the development process they are, naturally, easier to fix. Regarding autonomous driving, high-quality test data management, and hence data exchange and data fusion, is key to facilitating interoperability between systems, manufacturers, technical services, and other stakeholders involved.

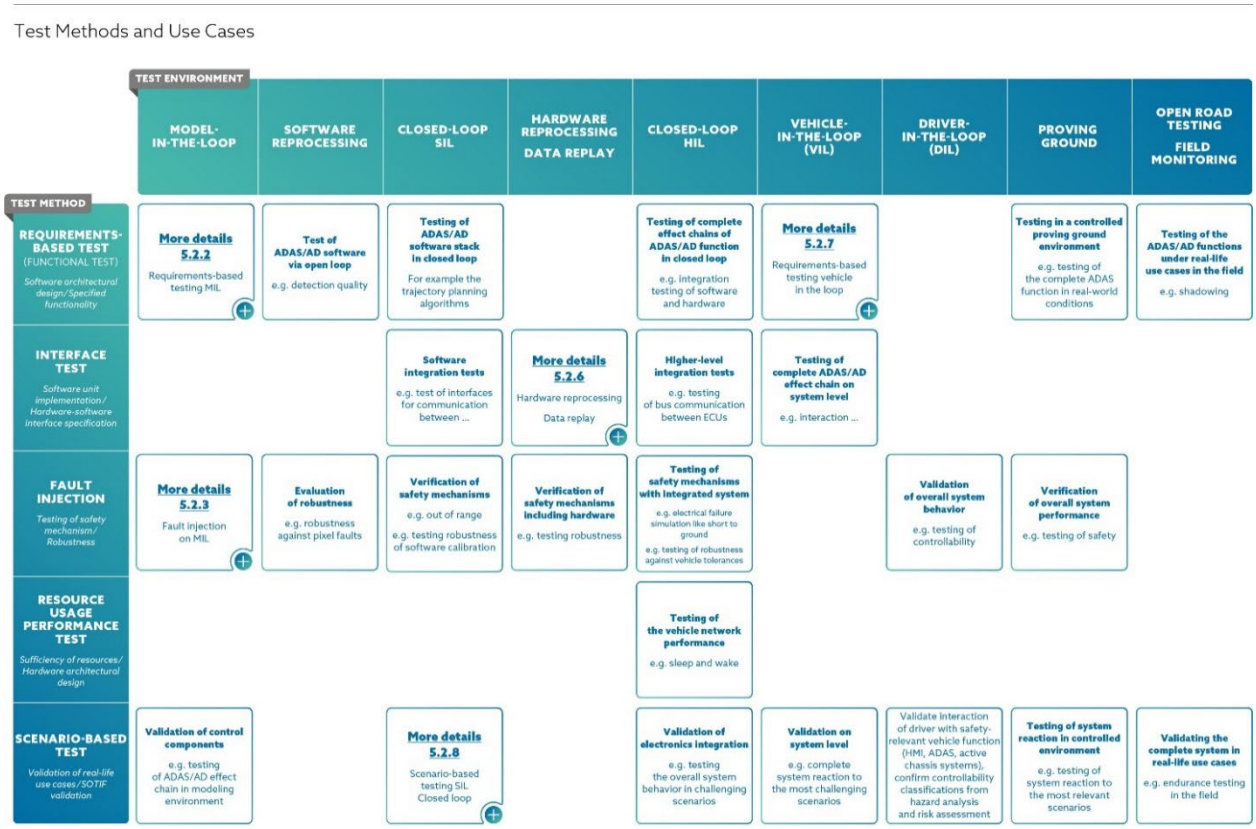
Motivation for Investigation of the Test Data Management Aspect

The development of autonomous driving functions can be described as data-driven. In all stages of the validation, huge amounts of already recorded test data, scenarios, and test cases are necessary to investigate the behavior of the function under test (FuT) in different simulations and real-life tests. Therefore, vehicles and test rigs have to collect huge amounts of test data, such as imaging and bus data, while driving on real and simulated roads.

Looking at the importance of having test data and test descriptions available throughout the overall development, simulation, and real testing makes it understandable that key to an efficient development and validation process is an efficient scenario, test case, and test data management.

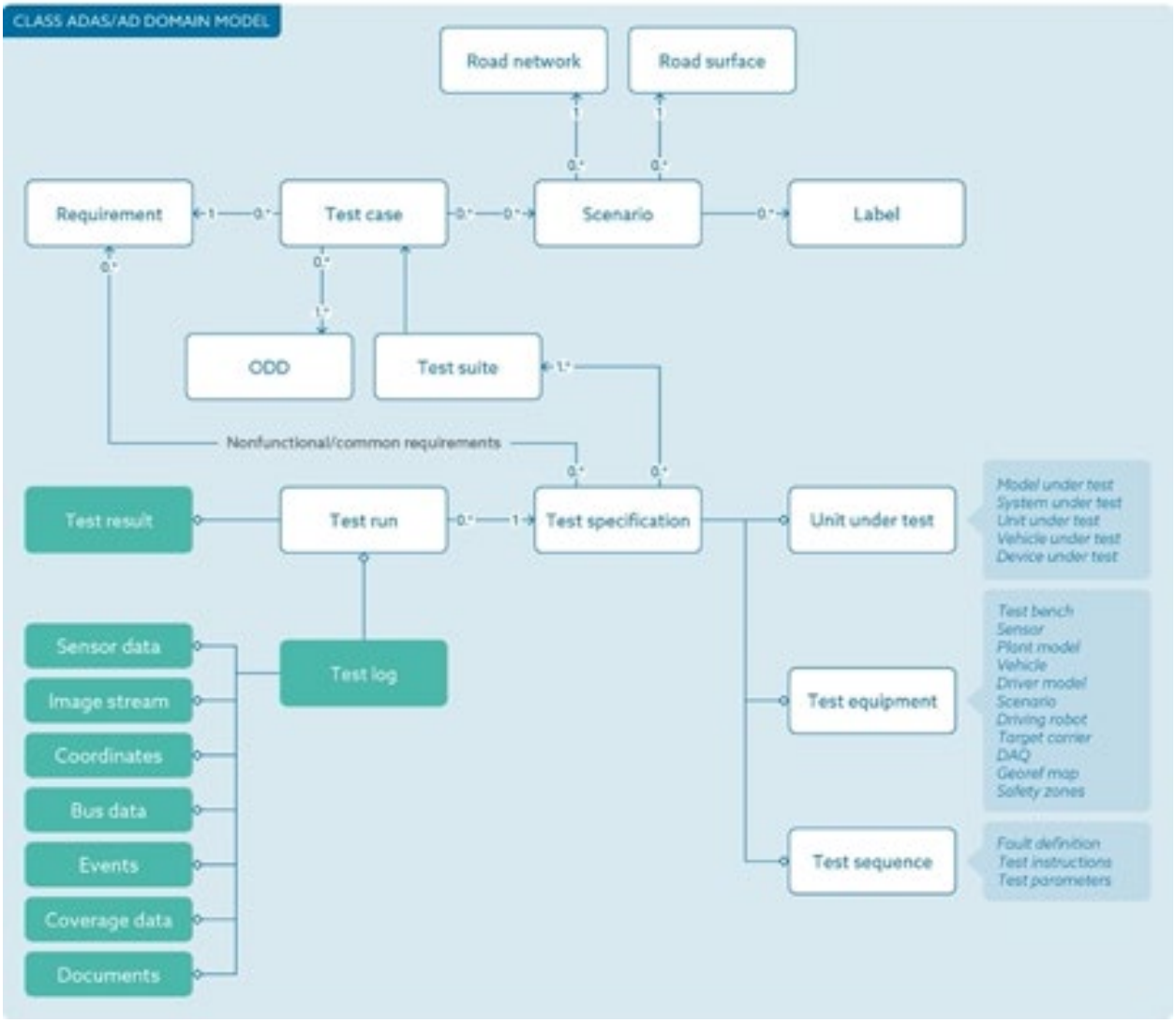
6.1 Data Modeling Based on Test Strategy and Use Cases

To describe test data management aspects in ADAS/AD function development we propose a first high-level domain model as a basis for furthering modeling in future standardization efforts.
The Test Strategy Blueprint and the according Use Cases defined their serve as bases as these are valid for the initial development process of ADAS functions until homologation.



To develop the high-level ADAS-specific domain model as the basis for proper test data management the use cases were investigated to find important artifacts. These could be test inputs, test outputs, or information necessary to assist the test preparation. Based on these findings the group developed an abstract “ADAS/AD domain” model, bearing the aspect of test data management in mind. The artifacts were displayed graphically and grouped according to their relation to one another.
The diagram distinguishes graphically between information which is necessary to (re-)conduct a test run, marked in green, and information which is created during or after a test run (marked in orange). It is based on the assumption that test specifications are derived from requirements. A description of the displayed artifacts is presented below.

Graphical Display of the Abstract ADAS/AD Domain Model



Artifact	Description
Test case	The different abstraction levels of a test case (logical down to concrete) are represented in the diagram through a reference to itself
Bus data	Describes in this context the recorded bus signal which was sent during the test execution
Test run	The execution of a test suite on a specific version of the test object (ISTQB)
Coordinates	In this context describes the movement of one or several test objects through a relative (e.g. proving ground) or global (GPS) coordinate system

Coverage data	The degree to which specified coverage items have been determined or have been exercised by a test suite expressed as a percentage (ISTQB)
Documents/reports	Final reports in different formats might be stored attached to tests, as it has been described in several testing workflows
Events	Singular moment in time at which some perceptible phenomenological change (energy, matter, or information) occurs at a certain place (ISO 21448, ISO 24765:2017, 3.1484). Trigger event: Event that causes an operation of a function (ISO 24765:2017, 3.4386 ISO 21448)
Image streams	Includes image-driven sensor data, e.g. lidar, radar, video data
Label	Set of labels which describe a scenario and the objects in the scenario (openLabel)
Operational design domain	Description of the specific operation domain(s) in which an automated function or system is designed to properly operate, including but not limited to roadway types, speed range, and environmental conditions
Requirement	A condition or capability needed by a stakeholder that must be met or possessed by a system, product, service, result, or component to satisfy a contract, standard, or specification. Or another formally imposed document to solve a problem or achieve an objective (ISO 24765:2017, 3.3431)
Road network	Description of the geometry of roads, lanes, and objects, such as road marks on the road, as well as features along the roads, like signals (ASAM OpenDRIVE®)
Road surface	Description of the road surface used in the road network description
Scenario description	Reference to chapter 5.3.9.1 Scenario Definition
Sensor data	In this context it relates to the “classical” sensor data related to physical dimensions (e.g. pressure, acceleration, voltage, etc.) as it is used in non-ADAS/AD test fields
Test description	Includes all descriptive elements which describe the execution of a concrete test. The test description includes all information about the used test sequence, test equipment, and unit under test
Test result	Indication of whether or not a specific test case has passed or failed, i.e. if the actual result observed as test item output corresponds to the expected result or if deviations were observed
Test sequence	Contains information about which sequence of steps has been processed during testing (ASAM ODS).

	In the reference workflows the following related artifacts have been found: • Fault definition • Test instruction • Test parameters In a future standardization project the description of these and further related information shall be standardized
Test equipment	Contains information about which equipment has been used (ASAM ODS) In the reference workflows the following related artifacts have been found: • Test bench • Sensor • Plant model • Vehicle • Driver model • Scenario • Driving robot • Target carrier • Data acquisition systems • Georef map • Safety zones In a future standardization project, the description of these and further related equipment shall be standardized
Test suite/campaign	A test suite is a group of test cases. A test campaign is the realization of an entire test strategy
Unit under test	Contains information about whether what has been tested (ASAM ODS) matches the test object in ISO. In the reference workflows the following related artifacts have been found: • Model under test • Function under test • System under test • Unit under test • Vehicle under test • Device under test

6.2 Impact on Homologation and Type Approval Process

Parallel to the shift to data-driven development and software-defined vehicles, the approval and homologation of automated driving functions is also changing. In the past, homologation was the last step during or after development. Here, subsets of tests were undertaken representing the identified hazards at vehicle level to check the overall system behavior. Due to the complexity of the driving task and subsequently of the systems, however, this is no longer sufficient, as the residual risk of such a highly automated system can no longer be determined at the overall vehicle level.

The reasons why comprehensible and uniform test data management is needed have become sufficiently clear. On the one hand, test strategies are becoming more complex and holistic, and on the other hand, the data required to correctly evaluate the results of the test strategy are more diverse.

For a comparable reason, type approval and homologation are also more complex and must rely on test or analysis results achieved based on the test data management, and on the proper implementation of the test data management.

The homologation of automated driving functions aims at proving the safety of the driving functions, as already presented in chapter 5.1. These criteria include proof that the driving function performs the driving task independently within the respective defined operating range, and that it complies with traffic regulations and it is able to autonomously switch into minimum risk conditions. As the test strategies described in this report represent one pillar necessary to approve all this, it is at the same time necessary to make this evaluable and assessable so that technical services can audit this. Consistent data management is indispensable for this. This report describes one possible approach.

6.3 Proposal for the Technical Application in the ASAM ODS Standard

Considering the flexibility of the ASAM ODS standard for different domains and its interoperability with the MDF standard (see section 4.3.5.1) and the need for the navigation of test results during ADAS/AD development and homologation (see previous section), we recommend an extension of the ASAM ODS standard by way of an ODS associate standard. This associate standard describes an ADAS/AD domain application model based on the abstract ADAS/AD domain model described. In this context the relations to other ADAS/AD-relevant standards also need to be defined (e.g. ATX, OTX, OpenSCENARIO, OpenXOntology).

The specific advantages of such an ADAS/AD-specific application model are presented below.

Advantages of developing a specific ASAM ODS – ADAS/AD application model

- Reusage of an existing, mature, and widespread standard for the management of a variety of data related to the validation of ADAS/AD, including metadata, time series data, bus data, streaming data, etc.
- ODS has proven in other automotive applications than ADAS/AD that it can enable traceability
- ODS provides enough structure and flexibility to adapt to different ADAS/AD-specific data storage requirements
- Standardized API for different tools in simulation to find and download scenarios, test cases, and recorded data (e.g. for reprocessing)
- Standardized usage and exchange of ADAS/AD test data throughout the industry (OEM, suppliers, test service providers, test and development tool providers, local and global authorities, etc.), e.g. when
  • Ordering internal and external test/simulation execution
  • Providing packages for homologation
  • Analyzing test and simulation results across department/company boundaries
  • Reusing recorded and calculated test data sets to derive new insights or as the basis for software and hardware reprocessing

- Possible creation of a comprehensive repository as a foundation for the data-driven development of ADAS/AD functions
- Existing format for exchanging model and test data (ATFx)

6.3.1 Preview of an ODS application model

The intention here is not to pre-specify but to give an impression of how an ADAS domain model could be represented in an ASAM ODS application model based on the available information so far. This model:

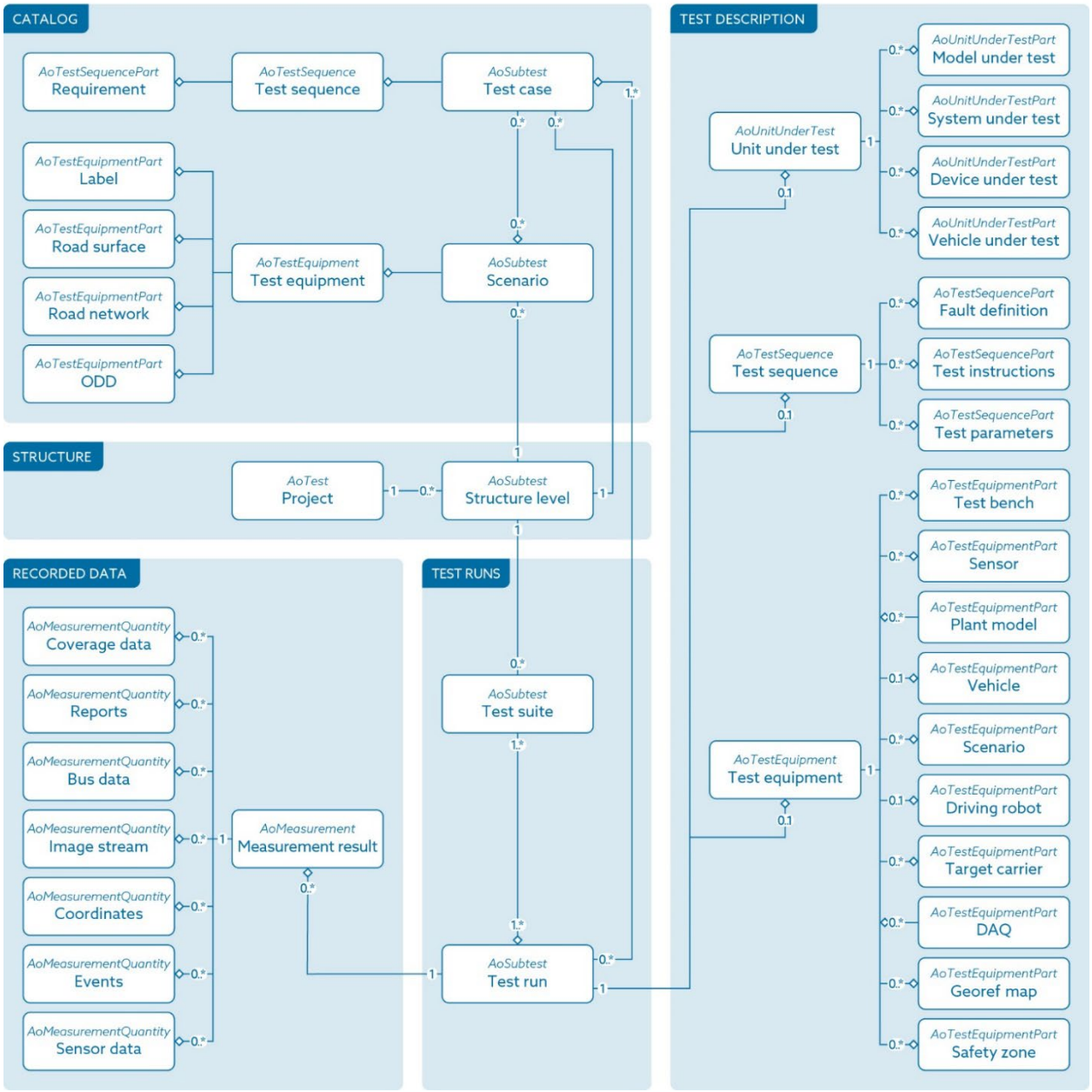
- Represents the artifacts found so far

- Allows for a catalogue of scenarios
- Allows for a catalogue of test cases
- Does not investigate the detailed information of each artifact

Different modeling approaches are possible. The model below, for example, is created with the use of the existing open-source framework openMDM in mind.

Most of the described artifact elements of the ASAM ODS base model have been temporarily assigned (e.g. AOSubtest for test campaign) until further discussion changes that idea. Others have been left empty, e.g. “Requirement,” to be specified at a later stage.

Example of a Possible ODS Application Model



In the following list only the main areas of the model are described.

Area	Description
Structure	This area allows items based on AOTest and AOSubtest to create a structure for navigation
Catalog	The catalog area consists of items which help to save a scenario catalog / test case catalog in an ODS storage. As they in many cases will also be stored externally, this area is optional for an ODS ADAS/AD repository

Test run	This area structures and saves all executed simulations and executed tests. With relations to recorded data and test description it provides all necessary information about the simulation/test for interpretation or repetition. https://glossary.istqb.org/en/search/test%20run (Test Log)
Recorded data	All recorded or arisen test data are saved and relate to a measurement
Test description	The test description area contains all entities which are necessary to describe all related metadata of a concrete test but can also save additional information, e.g. the used openScenario file, test automation file, model files, etc.

A future sub-standardization group could take this application model to a further level and develop a full application model similar to the NVH (noise vibration harshness) or passive safety domain.

7 Recommendations

The ASAM Test Specification Study Group project examined relevant techniques and use cases for testing and homologation of software-centric vehicles and automated driving functions. Identifying relevant standards, potential workflows, their variants, and the interplay between these parts led to a documented set of use cases, as well as to a set of potential workflows to implement them. Based on these findings, the study group developed a blueprint for test strategies that are ready for use by stakeholders across the automotive industry and across development phases. Moreover, the test-specification study group conducted a thorough white spot analysis, pointing out workflow gaps to identify where to supplement existing standards. The following recommendations are therefore the product of a joint effort by participants representing industry practice from various angles. Their shared goal is to establish a common pool of knowledge for further research, exchange, improvement and alignment. The bigger picture? A vehicle on the road and safety for everyone.

The New Test Strategy Blueprint as a Starting Point for Future Collaboration
The ASAM Test Specification Study Group is introducing a blueprint through this publication designed as a first point of reference. It was comprehensively developed with the implicit aim to be used, extended, or changed according to technological progress and testing possibilities. The recommendations linked to it are first and foremost an invitation.

Recommendations for a New Test Specification Standard
Scenario-based testing is highly popular and relevant for the safety argumentation, especially for automated vehicles. However, it is very clear from the report that this is only one part of testing. For a holistic release of the vehicle, its software, and electronics, a holistic approach as described in the Test Strategy Blueprint is necessary. To ensure that the different test methods and test environments along this test strategy are applicable and used, and firstly to use synergies, secondly to simplify transitions, and thirdly to increase the reusability of artefacts in all these tests, the study group proposes a standardized interface between a scenario description and a test case. It is recommended that a working group be set up to define and specify this interface.

The following aspects should be addressed:

- Depending on the use case there are test cases without any scenario description or test cases using one or even multiple scenario descriptions
- Different scenario description languages/standards exist but they need to be able to be connected to tests in flexible and common way.
- For the test description, different languages/standards are applied.
- Features of the “standardized interface” are driven from the test perspective; the scenario may not need to “learn” something from the test and vice-versa. Testing “drives” the

requirements on this interface and scenario descriptions implement such an interface (handshake).

It is important to emphasize that different standardization domains should not be mixed up by addressing them with a common standard. This holds especially true for the two domains of automotive testing and scenarios, where the above-mentioned interface between tests and scenarios should therefore separate the associated domains while also enabling their interaction.

Test Data Management as Basis for Release and Homologation

It has become clear in the course of the analysis that data management is necessary for the organization of the complex test landscape. In the course of the study, we were able to clearly demonstrate that the end-to-end testing and homologation of vehicles can be usefully supported with such test data management. However, we were not able to offer any analysis beyond the details of the workflows presented here. For example, topics such as traceability have not yet been sufficiently considered. The recommendation of the ASAM Test Specification Study Group is quite clear: to now follow up with a deep dive into the topic of data management and the associated (already existing) standards. For recommendations regarding specific of use cases or individual standards, please refer to the corresponding subsections.

Proposals for Global Alignment

As shown in the report, the map of testing is becoming increasingly complex and efforts to view these different activities holistically are also very heterogenous. This report serves as an invitation to analyze the different activities together and to look at them holistically. The clear recommendation is therefore to bring about a worldwide coordination of standardization in continuous follow-up activities. We currently see no tendency to reduce and unify standards and research projects. Therefore, our opinion is to invest more in alignments. The goal should be to answer the following question:

How can standardization be managed globally in the future? How long will we be able to accept this uncontrolled spread?

The automotive landscape is evolving but so is how we collaborate. Let’s put it to the test.