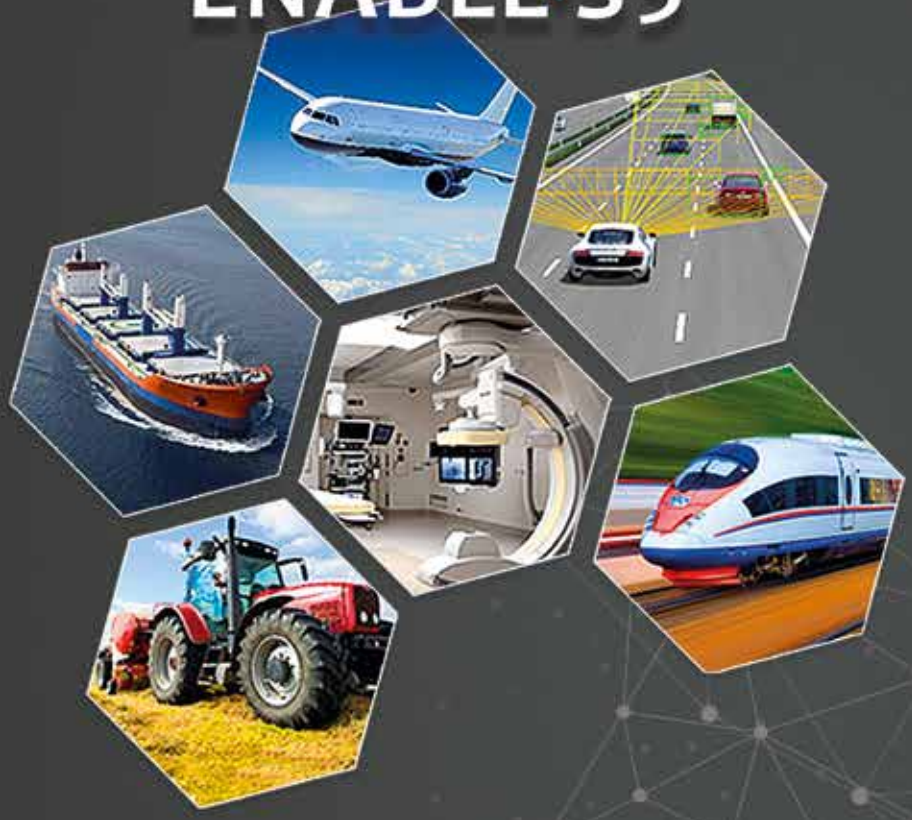
ENABLE-S3



TESTING & VALIDATION OF HIGHLY AUTOMATED SYSTEMS

Summary of Results

May 2019

This document summarises the key results of the European research project ENABLE-S3, an European initiative to ena**b**le the va**l**idation of highly automat**e**d **s**afe and **s**ecure **s**ystems.

68 industrial and academic partners from 16 European countries worked on methods and corresponding tools to be used to verify and validate automated vehicles, remotely controlled vessels, airplanes taxiing at airports, automated farming harvesters, trains as well as automated operating devices in the health domain. As these topics are of large interest to industry and at the core of various industry initiatives, the ENABLE-S3 project team intends to ensure that the ENABLE-S3 results will be applied successfully beyond the current number of project partners.

In this document, the project work and key findings of the ENABLE-S3 project have been summarised, defining and demonstrating a practical, cost-efficient way to verify and validate highly automated systems, building the basis for future product releases and certification/homologation processes. The intended audience of this document are research- and engineering managers, public funding authorities as well as regulatory bodies to get an overview of recent advances in the field of scenario-based verification and validation for highly automated systems.

The ENABLE-S3 research project was supported by the ECSEL Joint Undertaking of the European Commission as well as by the national funding authorities of the project partners. We would like to thank each participant for their contribution, making it possible to bring together experts from industry, academia and research institutions and creating a verification and validation framework for highly automated systems.

We would like to express our sincere thanks to the authors and reviewers of this document:

Andrea Leitner, AVL List, Austria
Arnold Akkermann, OFFIS, Germany
Bjørn Åge Hjøllo, Navtor, Norway
Boris Wirtz, OFFIS, Germany
Dejan Nickovic, AIT, Austria
Eike Möhlmann, OFFIS, Germany
Hannes Holzer, Virtual Vehicle, Austria
Jaap van der Voet, Philips, Netherlands
Jürgen Niehaus, SafeTRANS, Germany
Mathieu Sarrazin, Siemens, Belgium
Marc Zofka, FZI, Germany
Martijn Rooker, TTTech, Austria
Martin Kubisch, Airbus, Germany
Michael Paulweber, AVL List, Austria
Michael Siegel, OFFIS, Germany
Mika Rautila, VTT, Finland
Nadja Marko, Virtual Vehicle, Austria
Peter Tummeltshammer, Thales, Austria
Philipp Rosenberger, Technical University Darmstadt, Germany
Relindis Rott, Virtual Vehicle, Austria
Stefan Muckenhuber, Virtual Vehicle, Austria
Sytze Kalisvaart, TNO, Netherlands
Thies de Graaff, OFFIS, Germany
Thomas D'Hondt, Siemens, Belgium
Tobias Fleck, FZI, Germany
Zora Slavik, FZI, Germany

Also, without the assistance of Sarah Woywod, who coordinated the writing activities, revised and edited this document, it would not have been possible to complete the manuscript. Many thanks for your dedication.
We would also like to show our gratitude to all partners for participating in this project and for sharing their experiences. We are grateful for the MOBILITY.E Liaise, which brings together the experts working on burning topics in the automotive industry as well as all from other domains. May this document help to spread the results of ENABLE-S3 so they can be used in successful industry projects and future research activities.

Andrea Leitner
Project Coordinator ENABLE-S3

Michael Paulweber
Director Research AVL List GmbH

TABLE OF CONTENTS

INTRODUCTION

Increasing automation of cyber-physical systems – as in self-driving cars or ships, automated assistance systems in medicine and similar – is a major contribution to overcome many of the major societal challenges caused by a changing world with an ageing population that is living more and more in urban environments. Increasing automation in these systems will help the preservation of natural resources, air quality, clean and efficient transportation, new infrastructures, efficient and cost-effective health care solutions[1] , and many more, all to improve the quality of life. Affected application domains range from mobility (automotive, aerospace, rail, maritime) to agriculture and health-care. They all need to introduce safety-critical automated systems to overcome these challenges.

Although several technology demonstrators for highly automated systems already exist, there is a severe lack of cost-effective, commonly accepted verification & validation (V&V) methods and the need for tools supporting these methods. Winner et.al.[2] for example predict that more than 100 million km of road driving would be required to statistically prove that an automated vehicle is as safe as a manually driven one, which implies that a proven-in-use certification by performing physical tests on the road only is simply not feasible any more. Similar statements hold for other application domains as well. This lack of effectively applicable V&V methods created a severe barrier for the market introduction of these systems. The major challenge in all domains is the tight interaction of these safety-critical systems with their environment. This means that not only the correct functioning of the automated cyber-physical system itself should be tested, but also its correct reaction to the behaviour and specifics of its surroundings. This leads to a huge number of potential scenarios every automation system has to handle in a safe way.

The aim of the ENABLE-S3 project was to provide the required means for the verification & validation[3] of automated cyber-physical systems (ACPS). In a nutshell, the solution pursued in ENABLE-S3 is the identification of relevant scenarios, the automatic derivation of manageable sets of test cases from scenarios as well as the application of automated virtual V&V approaches in combination with physical test – in summary called scenario-based V&V of ACPS. In ENABLE-S3, industry and research partners from different application domains (automotive, aerospace, rail, maritime, health and farming) have joined their forces to develop the required technology bricks for the virtual V&V of automated cyber-physical systems.

A consortium of 68 partners representing OEMs, tier 1/2/3 companies, tool suppliers, academia and research centres along the value chains of the different domains applied for the ENABLE-S3 project in the ECSEL joint undertaking program of the European Commission. The proposal was funded and was executed successfully from May 2016 - May 2019 with a budget of €68 Mio.

[1] *http://ec.europa.eu/research/participants/data/ref/h2020/other/legal/jtis/ecsel-multi-stratplan-2018_en.pdf*
[2] *Winner, Hermann, and Walther Wachenfeld. "Absicherung automatischen Fahrens, 6." FAS-Tagung München, Munich 2013.*
[3] *Verification answers the question: "Am I building the thing right?", while validation answers the question "Am I building the right thing?"*

The ENABLE-S3 partners agreed on a **scenario-based verification & validation (V&V) approach, also advocated by projects like PEGASUS and the use cases of the ENABLE-S3 project described in the Demonstrator Overview booklet of the final ENABLE-S3 event.** This means that the main testing effort is shifted to a virtual environment largely represented by models. This has several advantages: tests can be conducted much earlier, cheaper, safer, and in a reproducible way. The inputs for the testing process are scenarios that the system under test (SuT) may encounter in the real world and need to be able to deal with. The term "scenario" is described in more detail in chapter 1.1.1 (Test Framework).

Because of the diversity of the project partners and the application domains, the project did not aim for a single common, generic software solution. The idea pursued was the **development of a common methodology, a basic verification and validation tool chain architecture and a set of reusable technology bricks** (tools, methods, models, etc.), which can be used to build up a testing environment for use case in different industry domains. Figure 1 1 illustrates the main structure of the project for the development of a modular framework for verification and validation of automated cyber-physical systems.

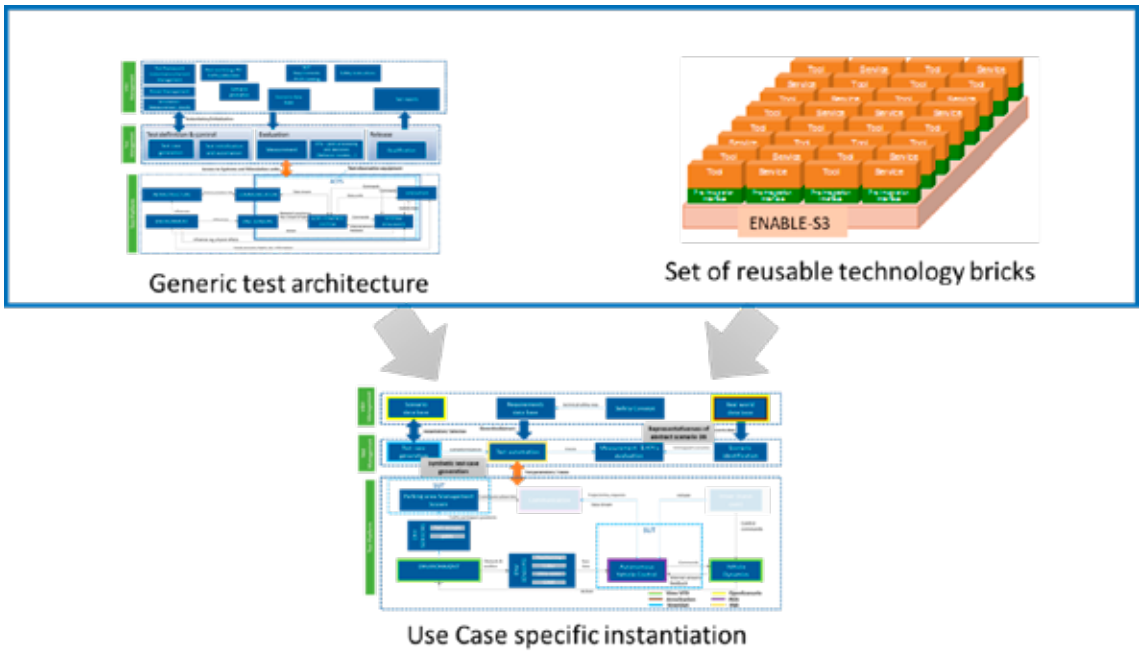


Figure 1 1: ENABLE-S3 development approach

Because of the large scope and complexity of the problem, it has been split into two parts as highlighted in Figure 1 2. The verification and validation methodologies on the one hand describe the necessary steps and research on data acquisition and storage, scenario and metrics selection, as well as test generation methods. There is a huge number of potential scenario classes that are either extracted from recorded data (real-world data) or generated synthetically (e.g. based on safety and security analysis). In real-world, a lot of variations for these scenarios exist (i.e. for different environmental conditions, different persons/traffic participants involved, etc.) leading to an enormous number of scenario instances. The goal is to provide intelligent methods to select the required scenarios instances in a way that ensures sufficient test coverage and to decide, which test cases should be executed in which test environment.

The V&V platform focuses on reusable technology bricks (tools and models), which can seamlessly support various development and testing environments (model-in-the-loop, hardware-in-the-loop, system-in-the-loop (e.g. vehicle-in-the-loop), as well as real-world testing). By combining both parts and their respective technology bricks the project aims for a significant reduction of the required test effort or even to enable testing of highly automated cyber-physical systems at all, respectively.

1.1 Generic Test Architecture

As mentioned before, one major goal of the ENABLE-S3 project is to deliver reusable technology bricks and seamless development environments, which can be used to build V&V tool chains implementing the workflow in Figure 1 2. The first promotes the development of models and tools that are easily reusable in different contexts. The latter requires to set up a testing environment where virtual representations can easily be exchanged by physical components. For both, the use of a modular structure with well-defined interfaces is essential. This resulted in the definition of a generic ENABLE-S3 test architecture as shown in Figure 1 3

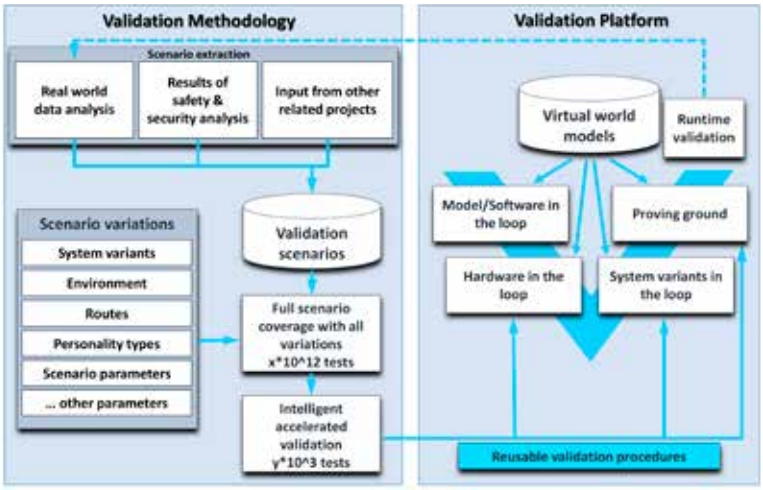


Figure 1 2: ENABLE-S3 validation tool chain architecture

This architecture aims for supporting the integration of different technology bricks in a concrete test system instance. It consists of three main layers and includes the most essential parts for testing automated cyber-physical systems (ACPS). The architecture is independent of the application domain and has been used for the V&V of industrial use cases from all six ENABLE-S3 application domains (automotive, aerospace, rail, maritime, health care and farming). The concrete characteristics of the blocks depend on the specific use cases. For some use cases, the blocks might be interpreted slightly different or are not required at all.

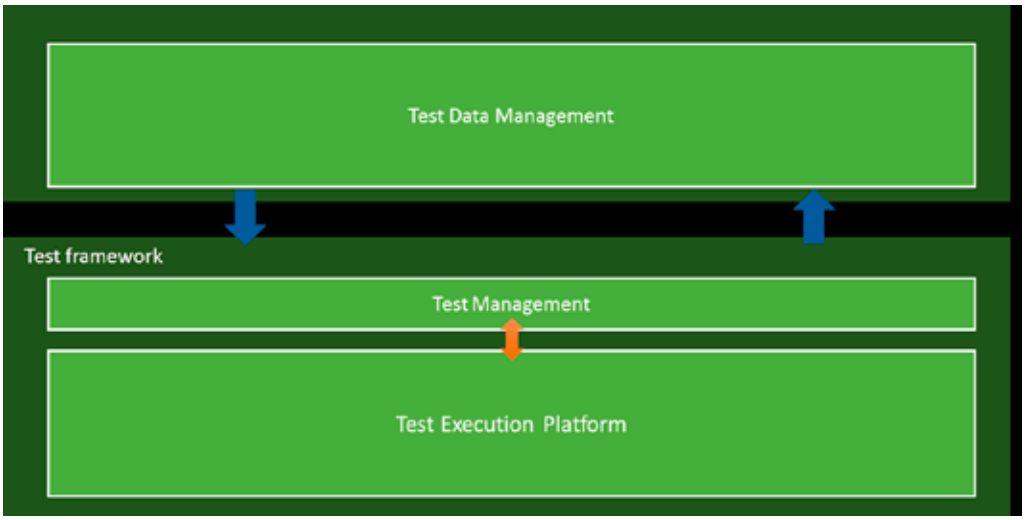


Figure 1 3: Main building blocks of the ENABLE-S3 generic test architecture

On a high level, we distinguish between the *Test Framework* and the *Test Data Management*. The Test Data Management covers all aspects which are valid across test phases and are reusable for testing different products. The *Test Framework* summarizes all aspects required for the planning (*Test Management*) and execution of tests (*Test Execution Platform*).

1.1.1 Test Framework

The test framework is divided into two parts: *Test Management* and T*est Execution Platform* (Figure 1 4). The main aspects are described in more detail in the following section.

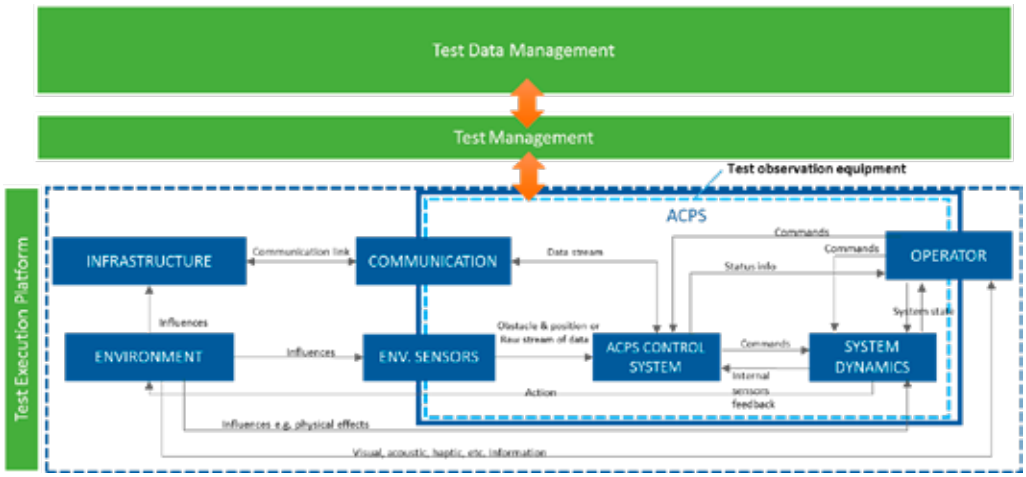


Figure 1 4: Reference architecture for test execution

Test execution platform

The *Test Execution Platform* covers all relevant aspects for testing an automated (cyber-physical) system including the SuT (either as a model, as software component, as subsystem or as complete system). The automated (cyber-physical) control system interacts with its environment (e.g. driving on a road, which is shared with other traffic participants, etc.) using d electronic sensors and embedded software. For the interaction, the automated (cyber-physical) control system has to perceive its environment either via sensors or the communication to the infrastructure or both. The system itself is described by its physical dynamics, which again needs to be fed back to the environment. The arrows in Figure 1 4 show the basic interactions of these testing architecture blocks. The concrete description of the interfaces depends on the application domain as well as on the concrete use case. For certain aspects, standardization of the interfaces is proposed (cf. Section 1.2).

Depending on the development stage of the SuT, there will be different instances of the test platform/architecture. For example, in a MiL (Model in the loop) environment all components will be available as simulation models. Later, simulated components will be successively substituted by real physical components resulting in a mixed environment of models and physical components.

In a MiL environment, the automated cyber-physical control system describes the main system under test (SuT). In later development stages, more aspects are integrated in the SuT (e.g. real sensors).

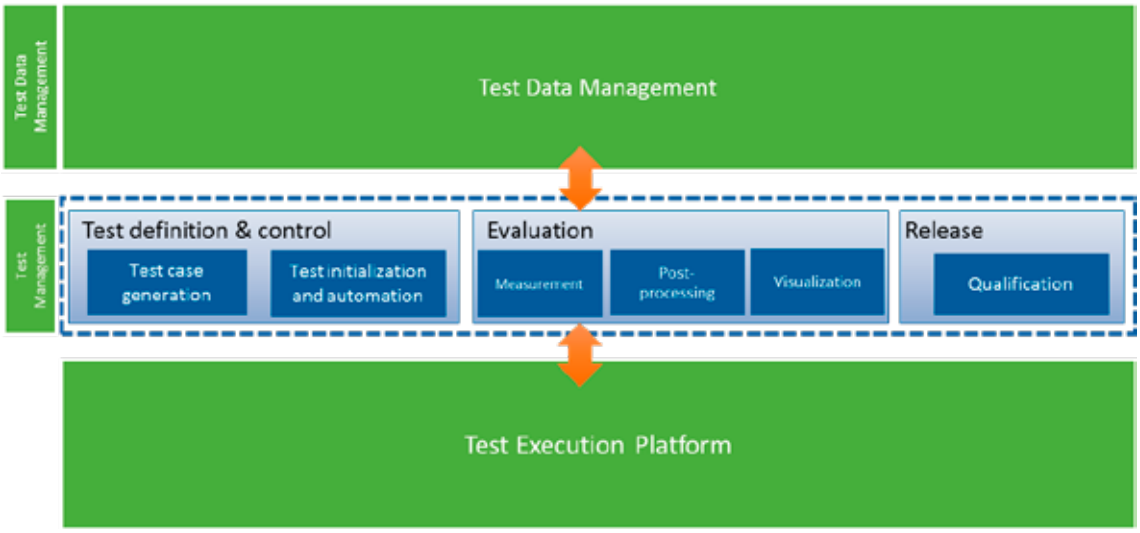


Figure 1 5: Test Management in more detail

Terms used in ENABLE-S3

Scenario class: A scenario class is a formalized description of the multi-actor process, including its static environment, its dynamic environment and environmental conditions. In a scenario class, the parameters are described and may have pa rameter ranges or distributions.

Scenario: A scenario is a formalized description of the multi-actor process, including its static environment, its dynamic environment and environmental conditions. In a scenario, the parameters are described and have fixed values. A scenario may include activities, events, goals of the activity and decisions of actors.

Explanation: a scenario is a unique instance of a scenario class. It may be ob served in the real world or created synthetically.

Activity: An activity refers to the behaviour of a particular mode of a system.

Explanation: For example, an activity could be described by the label 'braking' or 'changing lane'.

Event: An event marks the time instant at which a transition of state occurs, such that before and after an event, the state corresponds to two different activities.

Explanation: For example, an event could be described by the label 'initiate braking'.

Scene: A scene describes a snapshot of the environment including the scenery and dynamic elements, as well as all actors' and observers' self-representations, and the relationships among those entities.

Situation: A situation is the entirety of circumstances, which are to be considered for the selection of an appropriate behaviour pattern at a specific point in time. It entails all relevant conditions, options and determinants for behaviour.

Test case: A scenario selected from a broader scenario class with all values specified to be used for testing within the operational domain of a function under test.

Explanation: a test case is selected with a specific test purpose and operation al domain in mind.

Real-world scenario: Equivalent to scenario

Explanation: The term real-world shall indicate, that the scenario was derived from a real-world observation.

Synthetic scenario: Equivalent to scenario

Explanation: The term synthetic shall indicate, that the scenario was not derived from a real-world observation but rather crafted by hand, generated algorithmically or derived from a simulation instance.

Test scenario: Term to be avoided. Correct term: 'test case'

Test Management

Figure 1 6 shows the different aspects of the *Test Managemen*t in more detail. The blocks are assigned to 3 groups.

This layer covers the generation of a representative set of test cases from scenarios. This means that an interface to a scenario database is required to query the required information: relevant scenarios classes together with their variation parameters (e.g. weather, type of operator, type of route, equipment, etc.). Depending on the testing purpose this module has to include intelligent methods to select and instantiate the required scenario instances and prepare test cases. Then, the test cases need to be handed over to and executed in the T*est Execution Platform* (or more concretely in the Environment block of the *Test Execution Platform*). The

results need to be recorded, processed and potentially also visualized for inspection. In the last step, an proof for the overall safety of the system needs to be provided.

1.1.2 Test Data Management

This part (see Figure 1 6) focuses on all aspects that are valid across different test environments and includes the management of different types of data (i.e. measurement results, scenarios, etc.). It includes the establishment of a managed tool chain, which is also an important aspect in the long term – especially if virtual V&V environments shall be used for homologation/certification.

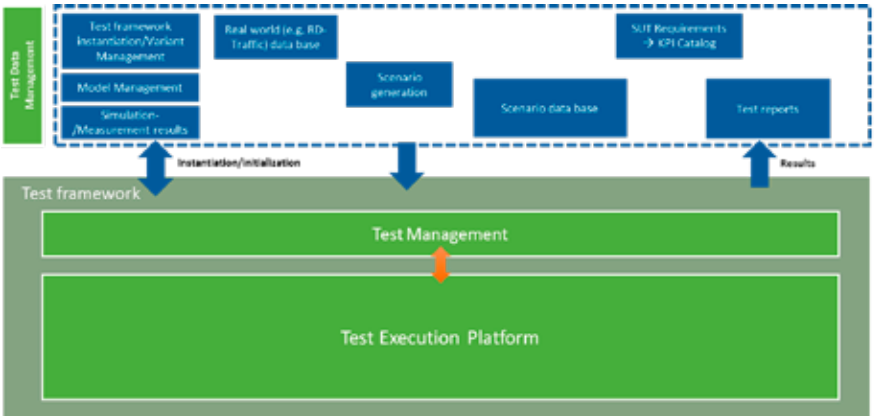


Figure 1 6: Test Data Management in more detail

The ENABLE-S3 project aims for a scenario-based verification and validation approach. A major prerequisite is the existence of a set of scenario classes which can be instantiated and executed. These scenario classes and their potential variation parameters can either be extracted from real-world data or generated synthetically by simulations. The “scenario generation” block summarizes all activities, methods and tools which are required to generate scenarios (e.g. by identifying and transforming critical real-world scenarios) which can be executed by an environment simulation engine. For traceability and reproducibility, it is further required to store all test artefacts and their interrelations.

1.2 Standardization Activities

To support the modular structure which is required for the applicability of the technology brick approach, standardization of interfaces is an important prerequisite. Therefore, a special focus has been put on standardization activities. A dedicated sub work package ensured a coordinated and efficient way to approach standardization organizations. In the following, some highlights are given. More details are provided in the respective subchapters.

1.2.1 Standardization of Scenario Descriptions

In the previous section, the scenario-based virtual V&V approach has been introduced. To make this practicable, scenarios (scenario classes as well as instances) need to be reusable in different environments and need to be shareable. This means that they need to be represented in a format, which is understood and interpreted in the same way by different simulation tools. For automotive, there are currently two open formats (OpenDRIVE®[4] and OpenSCENARIO®[5]) available, which – during the course of ENABLE-S3 and driven by ENABLE-S3 partners – are both now managed by the standardization organization ASAM eV[6].

OpenDRIVE®[7] is an already quite established specification for describing the logical view on the road networks (i.e. road curvature, lane information, speed limits and directions for single lanes). This specification is already supported by in prototype versions of various environment simulation tools such as VTD from VIRES/MSC, PreScan from TASS/Siemens or CarMaker from IPG.

Currently, the specification is restricted to automotive applications. Nevertheless, certain aspects and design decisions might be reused in other application domains (e.g. to describe routes for vessels).

OpenSCENARIO® is an open file format for the description of dynamic contents in driving simulation applications. The OpenSCENARIO-project is in an early stage and just starts to be supported by environment simulation tools. It is targeting the description of dynamic aspects of scenarios (i.e. traffic participants and their interaction). Again, the specification is currently developed for the automotive domain but might be adapted for other domains as well.

1.2.2 Standardization of Sensor Model Interfaces

The Open Simulation Interface[8] (OSI) is an upcoming standard to describe the data structure (message-based) of virtual perception sensors. It has been introduced by BMW and the Technical University of Munich and has been published as an open source project. This specification covers information like lidar point clouds or object lists, which are relevant as possible output of simulated perception sensors and sensor systems. Regarding the generic test architecture, OSI provides a standardized interface for perception sensor data, which is used by automated driving functions. Hence, this interface enables the connection between function development frameworks and the simulation environment. A standardized interface for the description of environment data is helpful to provide compatibility between different frameworks. More information is shown in Sec. 2.4.

[4] *http://www.opendrive.org*
[5] *http://www.openscenario.org*
[6] *https://www.asam.net*
[7] *http://www.opendrive.org*
[8] *Timo Hanke, Nils Hirsenkorn, Carlo Van-Driesten, Pilar Gracia-Ramos, Mark Schiementz, and Sebastian Schneider. Open Simulation Interface: A generic interface for the environment perception of automated driving functions in virtual scenarios: Research Report, 2017. http://www.hot.ei.tum.de/forschung/automotive-veroeffentlichungen/, (Accessed: 2017-08-28)*

1.2.3 Sustainability of Results

All specifications described above have already been available at the beginning of the project. Nevertheless, within the ENABLE-S3 project we identified these specifications as essential for the automotive domain. Therefore, we applied them and identified further requirements from our project use cases.

As a major result, the specifications have been handed over to the ASAM e.V. standardization organization and are now managed there in dedicated working groups. This will make sure that the results of the project are sustained, accessible and further developed after the project is finished.

1.3 Structure of the Following Chapters

This paper aims to give an overview on the key achievements and outcomes of the ENABLE-S3 project.

The document follows the structure of the projects' work packages (WP) as shown in Figure 1 7. WP1 and WP4 focus on the industrial use cases of the different domains. In WP1, the systems under test as well as the test systems have been specified. This resulted in a set of requirements, which have been handed over to WP3. There, the requirements have been harmonized across the application domains (wherever useful) and the required V&V technology bricks have been developed. To assess the developed solutions, they have been handed over to the application use cases for integration, demonstration and evaluation. After several iterations, this resulted in a set of reusable technology bricks as well as several industrial demonstrators. Chapter 2 highlights the key results of WP3 (technology brick development) and Chapter 3 summarizes the impact of the project results for the different application domains. Annex 4 gives an outlook on further research recommendations as an outcome of the project. WP2 has mainly acted as a support work package for requirement management, progress monitoring and defining the KPIs for measuring and assessing the project results. WP5 includes all activities regarding dissemination, exploitation and standardization. These topics will be mentioned in the context of the technical work packages in this paper and therefore not addressed separately.

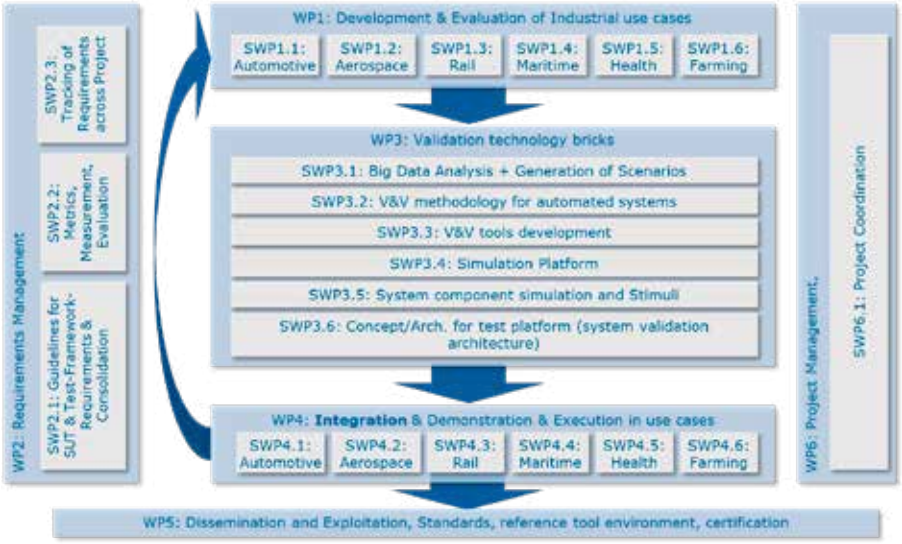


Figure 1 7: ENABLE-S3 Work package structure

Chapter 2.1 documents all activities related to **V&V methodology for automated systems.** It addresses modelling, simulation, testing, and coverage criteria and aims for maximizing synergies between the domains and use cases involved in ENABLE-S3. Chapter 2.2 smmarises the main achievements in the field of **Big Data Analysis and Scenario Generation.** It focuses on gathering available test data (e.g. scenarios from real-world data) as well as on the generation of new data wherever necessary. Chapter 2.3 lists the developed tools and respective tool extensions that have been implemented based on the methodology development. The **Simulation platform** described in Chapter 2.4 Simulation platform for system validation elaborates a cross-domain, co-simulation-based ACPS integration platform, supporting both non-real-time (offline) and real-time simulation as well as the required models (i.e. environment, driver, and ACPS components). Chapter 2.5 System Component Simulation and Stimuli finally highlights the achievements in the field of **System component Simulation and Stimuli**. This includes modelling and simulation of required sensors, such as passive sensors (e.g. cameras), active sensors (e.g. radar or ultrasonic), as well as vehicle-2-x communication or GPS.

2 KEY OUTCOMES OF TECHNOLOGY BRICK DEVELOPMENT

As indicated in Figure 1 1, one main outcome of the ENABLE-S3 project is a set of reusable technology bricks (tools, methods and models). These bricks have been clustered in work packages. The concrete developments are described in more detail in the respective deliverables. In this chapter, we will highlight the key achievements that were made in the ENABLE-S3 project.

2.1 Scenario-based V&V Methodology

Key Results:

- Development of a scenario-based V&V methodology that addresses the specific aspects of highly-automated systems, such as self-driving vehicles and smart medical devices, in a systematic manner. The proposed methodology takes a holistic approach and explains the steps from the initial understanding of the operational context until the final safety and security argument to release the highly automated system. This substantially enhances the previous V&V methodologies that were tailored to systems with little or no autonomy.

  - The developed scenario-based V&V methodology is generic and relevant for many classes of highly automated systems. In contrast to previous more domain-specific approaches, the methodology developed in ENABLE-S3 subsumes best-practices collected in 6 different application domains to which scenario-based V&V was applied.

- Scenario-based V&V provides the basis for the technical developments in the project and is the “glue” between other key outcomes described in the subsequent chapters

This chapter summarizes the key ingredients of the scenario-based V&V methodology for highly-automated systems, one of the major outcomes of the ENABLE-S3 project. The proposed methodology depicted Figure 2 1, identifies advanced methods and solutions that are tailored to address the complexity and specificities of highly automated cyber-physical systems, including their intricate system dynamics, unpredictable environments with possibly emergent behaviours, and defines an effective and efficient workflow based on best V&V practices. It explains the steps from the initial understanding of the operational context resulting in func-

tion and domain scoping, until the final safety argument to release the highly automated cyber-physical system.

This methodology results from the interaction between industry and academia and represents a high-level and generic workflow that is applicable to many classes of highly automated systems. It inherits the experience collected from 6 different application domains. This broad knowledge accumulated from various applications allowed us to distinguish domain-specific from domain-agnostic methods. Consequently, the proposed methodology provides a generic application-independent workflow that can be instantiated in a way that acknowledges specificities of a given domain. The methodology is the basis for subsequent technical project results (new methods, tools, etc.), providing the "glue" between the other key outcomes described in the subsequent chapters.

The central concept in our V&V methodology is the notion of a scenario (see Section 2.2.2). Scenarios enable identifying and exploring both common and critical scenarios that a highly automated cyber-physical system may encounter during its operation and that need to be tested. In addition, scenarios allow to characterize the environment and the context in which the highly automated system is being used. Scenarios are presented in more details in Chapter 2.2.

Another crucial aspect in the scenario-based methodology is the combination of *virtual* and *physical* testing. Complementing physical with virtual testing offers multiple advantages:

- Large number of scenario parameters can be efficiently explored by simulation, allowing to reduce the number of costly in-field tests by reproducing in the physical setting only the most relevant scenarios;

- It allows to test challenging and dangerous scenarios in a virtual setting without representing any risk to humans and/or equipment;

- Specific conditions in the environment, such as fog, snow, road quality, low visibility conditions, etc. that are difficult to set up and control in the physical world can be effectively reproduced and tested in the virtual environment.

Consequently, a scenario-based V&V methodology addresses the inherent complexity of highly automated systems by providing a systematic, yet scalable and efficient V&V strategy. It complements the Generic Testing Architecture (GTA), also developed in ENABLE-S3 and presented in Chapter 1.1.1. In contrast to the GTA, which focuses on the description of the testing and the V&V architecture, the proposed methodology defines the V&V workflow, detailing necessary steps for meeting complex V&V requirements and deriving the overall safety arguments. This chapter shows the static view of the methodology, in which arrows between the

major steps (see Figure 2 1) represent the input/output relations between the V&V activities, but not necessarily their interdependencies or order.

The static view consists of six main activities, which are described in more detail in the remainder of the chapter: The 1) *Function and Domain Scoping activity* provides an initial scope of function and target domain. The 2) *Requirement and Scenario Elicitation* generates and refines functional requirements as well as a scenario (class) database, which forms the central part of scenario-based V&V. After thinning out the scenario database in the 3) *Scenario Filtering*, only relevant and informative scenarios remain. The 4) *Assessment and System Development activity* bundles safety, security and usability assessment with the actual system development process. System development is accompanied by the 5) *Virtual and Physical Testing of System Qualities*, such as safety, security etc. Finally, the 6) *Establish (Overall) Safety Argument activity* ensures that the developed system is acceptably safe.

The chapter ends with several application examples, which illustrate the usage of the scenario-based V&V methodology.

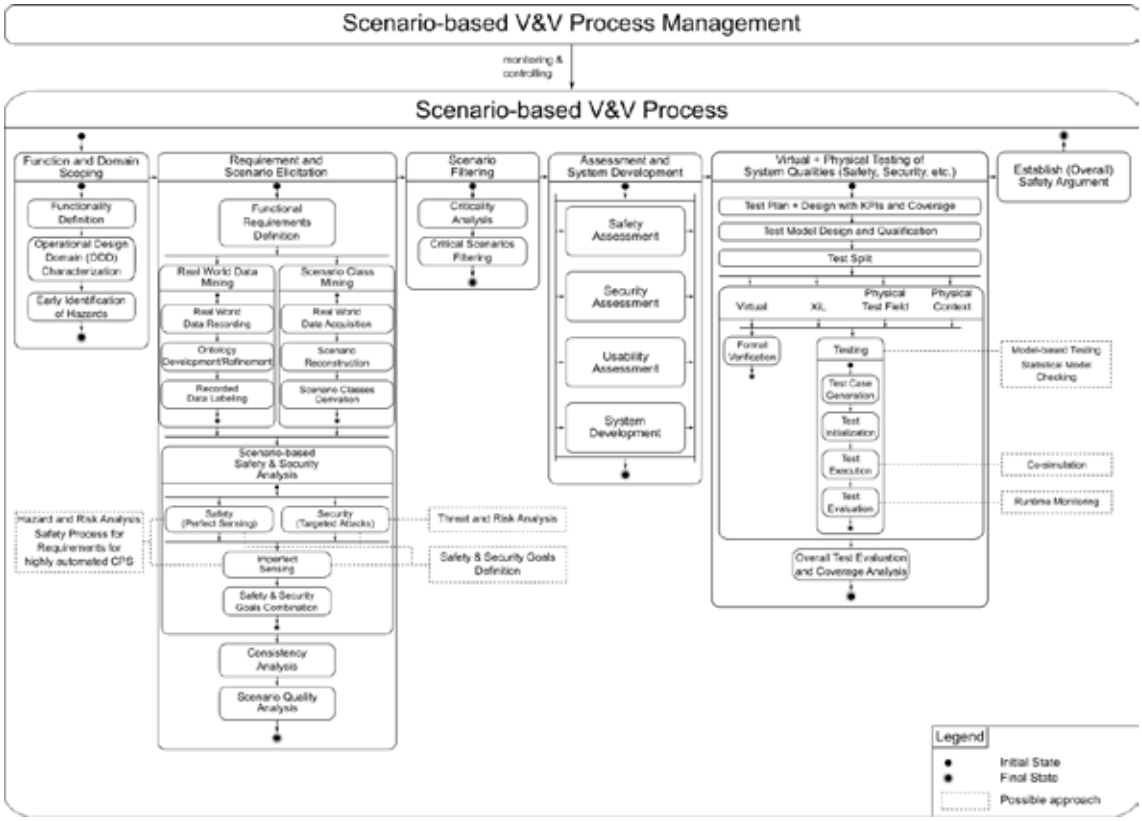


Figure 2 1: Overview of the scenario-based V&V methodology

2.1.1 Function and Domain Scoping

Scenario-based verification and validation starts with scoping of the function and the target domain. The function defines what to do (e.g. the primary driving task of an automated driving function). The domain (also called operational design domain [1]) defines in which context this function operates. At this point the function as well as the domain do not need to be formally defined but their description should be accurate enough to serve as the basis for the *Requirement and Scenario Elicitation* and the *System Development*. For example, in the maritime domain ENABLE-S3 worked on the validation of a maritime function that provides autonomous navigation and remote control of ships. To narrow down the number of scenarios to be validated, certain assumptions and restrictions are made, e.g. availability of communication infrastructure, all vessels are following the maritime traffic rules. This is important as it narrows down not only the number of scenarios that the navigation function should be aware of but also the number of scenarios that need to be covered during testing. It also gives a first estimation as to the conditions that must be met during the operation of the system in order to actually use the function. During the product development these assumptions and restrictions are iteratively weakened.

Functionality Definition

The result of this activity is an initial, rough definition of the function of the highly automated system, i.e., what should this system be responsible for. For example, the Highway Pilot System is intended to assist the driver in partly or fully automated driving along a pre-defined route. It should be able to drive fully automated on the highway, including entering and exiting the highway. To do so it has to be able to perform tasks such as speed control, distance control, accelerating and decelerating, lane keeping, object detection, overtaking and handle the hand over from manual driving to automated driving and vice versa in different scenarios.

Operational Design Domain (ODD) Characterization

The result of this activity is an initial characterization that is suitable for guiding the recording of scenarios, i.e., it allows judging whether a real-world observation belongs to the operational design domain and should therefore be considered as relevant for the highly automated system. For the highway pilot example, this could mean to define that the function should be able to operate on German and Austrian highways with up to three lanes and a physical separation between directions. It should be able to cope with different road markers, such as dashed and solid lines as well as orange (high priority) markers. It should be able to perform in various weather conditions, such as sunshine, rain and fog, but not in snowy conditions and the function should be able to deal with daylight and night time driving.

Early Identification of Hazards

The result of this activity is an initial list of hazards, i.e., harms that can be caused by the highly automated system during operation. Potential malfunctions are analysed and assessed according to severity, exposure and controllability. Severity represents expected injuries if an accident happens, exposure indicates how likely the scenario occurs, and controllability describes how controllable the scenario for the driver is. For the highway pilot function, a hazard could be that the function is unintentionally activated outside the operational design domain or that it is causing a collision with another traffic participant.

2.1.2 Requirement and Scenario Elicitation

The aim of the *Requirement and Scenario Elicitation* is to provide formal functional requirements, to perform scenario-based safety and security analysis, and to gather relevant scenarios and scenario classes.

Relevant scenario classes are systematically mined (*Scenario Class Mining*) from recorded real-world data (*Real-world Data Mining*) and complemented with additional scenarios uncovered during Scenario-based Safety & Security Analysis. The resulting requirements and scenarios are checked for consistency (*Consistency Analysis*) and quality (*Scenario Quality Analysis*).

Functional Requirements Definition

In this activity, the function description and the operational design domain obtained from the Function and Domain Scoping are refined and formalized. The resulting formal function requirements serve as basis for the Real-world Data Mining, the Scenario Class Mining, and the Scenario-based Safety and Security Analysis.

Real-world Data Mining

The goal of this activity is to gather semantically labelled data about the real-world environment (e.g., traffic scenarios encountered objects and their trajectories (if any), weather conditions), which can be used as base for extracting relevant scenario classes (see *Scenario Class Mining*). Such data may already exist (e.g. in traffic databases, accident databases, or OEM data collections), but also may be recorded specifically for this purpose.

In general, the execution time of the real-world data mining process may be loosely coupled to the scenario-based V&V process. It is possible that data gathering has been finished before starting V&V, it may be performed in the early process stages, or even may completely run in parallel.

Real-world data mining consists of the following sub-activities:

Real-world Data Recording: Real-world unstructured data is the basis for upcoming analyses and derivation steps. This data must be recorded (e.g. by equipping vehicles with appropriate sensors, traffic cameras, etc.) and the resulting datasets have to be entered into a real-world database.

Recorded Data Labelling: For being useful for scenario mining, it is not only necessary to have the data available, but it is as important to know what it means. Therefore, it is important that recorded datasets are provided with correct semantical labels.

This process can be performed with an automated pre-labelling process, however it should be manually checked and, if necessary, improved. Labelling should be carried out soon after data recording – the staler data gets, the more difficult it is to relate it to the world and the harder it is to generate adequate labels. The labelled datasets are entered into a labelled real-world database.

Ontology Development/Refinement: For the labelling of the real-world data it is necessary to develop an appropriate ontology, i.e. a systematic naming scheme for the relevant object kinds, the corresponding (inner) structures and the relevant relations. This development can be based on already recorded data. If an ontology is already defined, it can be checked for compliance to the recorded data. If necessary, appropriate adjustments to the ontology must be made (e.g. the recorded data contains an important entity, currently not represented in the ontology).

Scenario Class Mining

This is the process of acquiring relevant scenario classes from existing labelled real-world data. It consists of the following activities:

Real-world Data Acquisition: The first step for preparing data for scenario class mining is data acquisition. Datasets can be retrieved from already existing databases, if they are provided in necessary detail and completeness. For example, databases on traffic accidents may be a valuable source of information. But it is also possible to trigger the recording of new data specifically for that purpose (see Real-world Data Mining).

Scenario Reconstruction: For each labelled real-world dataset, a scenario is (re)constructed. The obtained scenarios are entered into a (rather extensive) real-world scenario catalogue with scenario instances that have been observed in the real world (see more details in Chapter 2.1.3).

Scenario Classes Derivation: The real-world scenario catalogue may contain many similar scenario instances (e.g. describing overtaking or lane changes). Multiple similar scenario instances are clustered into scenario classes. A scenario class is thus an abstract representation of scenario instances, capturing the shared essence of the corresponding real-world scenarios. Unnecessary details are stripped away, and some of the relevant details are replaced by abstractions and/or parameters.

Scenario class generation requires finding matching abstractions (abstraction levels). For instance, a general overtaking manoeuvre is a scenario class that abstracts a concrete manoeuvre (or a diversity of manoeuvres) that was observed and recorded on a highway. A class may abstractly describe properties such as scenery, weather conditions, and dynamic objects or even an abstract dynamic model which was also derived from the real-world scenarios.

The number of scenario classes is significantly lower than the number of scenario instances because each describes a multitude of instances. The scenario classes are collected in a scenario database.

Scenario-based Safety & Security Analysis

Scenario-based safety and security analysis aims at identifying safety hazards and security threats in order to analyse the corresponding risks.

Main result is the definition of security/safety goals along with the corresponding requirements and affected scenarios. The main difference to classical safety and security analysis is that in the context of highly-automated and autonomous cyber-physical systems hazards and security threats involve a much higher degree of dynamics than in classical contexts. I.e., the impact of a decision, a hazard, or a threat highly depends on the rich environment (the scenario) and the complex interaction of the system and other entities. Additionally, consequences might not be directly measurable but affect future evolutions. Therefore, the main task is to analyse the complex operational environment and how it affects the system under design.

If during this process scenarios are found, which are still uncovered in the abstract operational scenario database, then the database is extended.

A **scenario-based safety analysis** is used to identify hazards and corresponding scenarios in which this hazard can lead to harm. Here, one can apply the classic techniques such as *hazard analysis and risk assessment* (HARA) or *failure mode and effects analysis* (FMEA) but on the abstract function description as the one created in the Function and Domain Scoping and considering the beforehand identified operational scenarios. To simplify the process and to better locate issues, the analysis can – as a first step – assume perfect information. E.g. under the

assumption of perfect perception hazards are analysed that have to be dealt with at the manoeuver planning part of an automated function.

A **scenario-based security analysis** is used to identify attack vectors and the possible consequences of the identified attacks as well as the scenarios in which the attacks can occur. At this level the analysis should be performed on the abstract function description as the one created in the *Function and Domain Scoping*. Again, the security analysis should then consider the beforehand identified operational scenarios in that sense that only attacks that are feasible in the scenarios are to be explored and the identified countermeasures should prevent the attacks at least in these scenarios.

The STRIDE method was used for example in the "Touch And Go Assistant" use case of the aerospace domain. The idea is to look at each component of the system and consider if there might be threats from categories defined by the STRIDE method. At this stage one only looks at potential threats (without countermeasures), so a component which handles private data will always have the risk of information disclosure. As second stage one looks at the threats and how they affect the components and the interconnections with other components of the system. Finally, one looks at each threat and identifies countermeasures that prevent an attacker from abusing the system with the identified threats.

As the next step, assumptions initially made are relaxed. This is essentially repeating the prior safety and security analysis but this time focusing on additional impacts due to imperfect sensing. I.e., what could go wrong due to the inability of perceiving the surrounding world accurately?

Finally, the identified safety and security goals, i.e., the countermeasures need to be combined and further be broken up into (functional) safety & security requirements.

Consistency Analysis

In this step, it is checked that the collected requirements are consistent (they do not contradict each other) in all scenario classes. This can either be done by expert reviews or supported by tools if requirements are formalized.

Scenario Quality Analysis

As we will check whether a concrete *system under test* will satisfy the collected requirements (see *Virtual and physical testing of system qualities*) one has to make sure that the scenarios have a sufficiently high quality. Clearly, no scenario database (or a collection of scenario databases, each covering a specific aspect) will ever be complete with respect to all possible aspects, but it should be known for which requirements and for which parts of the operational design

domain the database is representative (i.e., experiments based on the scenario database will lead to the same or similar results experiments based on the real world) . In order to be able to make such an assessment, we need to be able to measure to which extent the database covers and accurately represents the real world.

To be more precise, one needs to identify for which operational scenario classes – which are subdomains of the operational design domain (for instance in the context of a parking garage, scenarios considering parking, but not general driving) –

1. real-world observations can be explained by virtual world observations,
2. vice versa i.e., virtual world observations can be mapped to real-world ones, and
3. real-world and virtual world observations have comparable frequencies, i.e. occur at roughly the same intervals.

As we are not interested in all possible aspects, one may only consider properties that are relevant for the collected requirements. For example, one might need to make sure that the distributions of velocities, turning rates, and accelerations are close for virtual world and real-world observations but other aspects such as colours, material, and sound can be ignored in case they are known to be not considered by the highly automated system or function under development.

Another important quality indicator for scenario classes is the risk of missing relevant scenarios or relevant aspects. For example, if handicapped persons moving in wheelchairs are missing in the collection of urban traffic scenarios, this collection of scenarios is only an incomplete representation of real-world scenarios. Here, the following should be considered:

1. bias introduced by the *real-world data mining and scenario class mining process*,
2. completeness measures, e.g., detection rates of new scenarios (see also chapter 2.2), and
3. systematic issues such as missing data for a certain sensor.

The results of this step influence the decision for the *Test split*, in the sense that those scenario classes for which the quality of virtual simulation is high, can be tested virtually, other scenario classes should be considered for mixed/physical testing.

2.1.3 Scenario Filtering

The basic idea is to focus on the relevant scenarios. However, as this is done on an abstract description of the SuT. Hence, one should rather (1) filter out scenarios which do not (or are unlikely to) bring insights, (2) filter out scenarios that are irrelevant, and (3) favour scenarios, which are more relevant than others for testing the satisfaction of requirements.

In the following we describe one way of filtering based on criticality.

Criticality Analysis

In this step, a database of so-called *criticality indicators* needs to be derived. These indicators define accident or near-accident scenario as well as violations and near-violations of the requirements that have been collected in the *scenario and requirement elicitation*. A typical example of a criticality indicator is the distance between two vehicles based on the requirement that two vehicles shall never collide. The scenarios may be derived from real world measurements or from requirements. A systematic way of collecting such criticality indicators is presented by Galbas and Damm[10].

Critical Scenarios Filtering

Next, the criticality indicators can be used to exclude scenario classes or instances of these for which accidents, near-accidents, requirement violations, or near-violations are not existent, or the remaining risk is below an acceptable tolerated level: This analysis may require simulation of the respective scenarios together with the system under test.

Further, the scenario classes may be annotated with those indicators, which can help in guiding the test in critical scenarios during the execution of the test case. I.e. by making the environment trying to reach scenarios which are specifically challenging for the system under test.

[10] *W. Damm and R. Galbas (2018) Exploiting Learning and Scenario-based Specification Languages for the Verification and Validation of Highly Automated Driving, in SEFAIAS'18, Gothenburg, Sweden, ACM, 2018.*

2.1.4 Assessment and System Development

Safety Assessment

During the safety assessment (as described in chapter 2.1.2 in the section "Scenario-based Safety & Security Analysis") the goal is to identify scenario instances with an impact on safety. Such an assessment ensures that a) all requirements are defined for the system development and b) the test phase covers all relevant scenarios to ensure that the system will operate safely in the intended environment. This is done by the safety analysis and the defined security goals and ensuring that all of this is included and considered in defining the test plan. Especially with the testing of safety critical scenario elements, virtual testing can offer a safe way to assess the system reaction.

Security Assessment

The goal of the security assessment is to identify if all security relevant aspects are considered in the scenario instances. This needs to consider a larger set of interactions and possibilities. While the scenario describes the operational, functional view and the environment, security represents an intentional and malicious outside influence. This means we cannot only rely on the scenarios derived from real-world data but that we also need to consider additional scenarios in which all relevant threat actors and potential interactions identified during the Threat and Risk analysis are integrated.

Here we have the option to test identified and repeatable malicious interactions as a first step in the virtual test and move then towards the real system with more open testing or penetration testing.

Usability Assessment

A common misconception is that usability can be designed into an automated system as a last step. In many automated systems, transition of control between user and automation is safety critical and complex. Therefore, it needs to be integrated into the system concept and safety concept from start.

The usability assessment evaluates the interaction between user and system to improve or assess the system. Given a realistic set of scenarios for functional tasks, system environment and system variants, the user is asked to perform the tasks. Success is measured with effectiveness (does the user reach the goal?), efficiency (effort to reach the goal) and user experience (how satisfactory is it to use the system?, how important is it to the user?, does the user identify with the system?).

In the context of the automotive domain, this could be done in real cars (normal traffic or test tracks) or in a driving simulator. This is depending amongst others on availability of the system to be assessed, safety concerns and requirements in terms of experimental control over scenarios.

Usability can cover various aspects of the system, ranging from the basics of the human-machine interface (are visual displays clearly visible and understandable, are buttons or controls easy to reach?, etc) to the operational behaviour of an automated driving function (e.g. in terms of comfort, etc). The usability assessment should ideally be done using both subjective measures (standardized questionnaires and/or interviews) and objective measures (regarding effectiveness and efficiency).

System Development

This task consists of building the system under test that implements its defined functions. The system is implemented at the right level of abstraction for the specific development stage. Consequently, system development can consist in building a virtual simulation model but also in building the physical system that will be put in the field for operation.

2.1.5 Virtual and Physical Testing of System Qualities (Safety, Security, Reliability)

Test Plan and Design with KPIs

A test plan uses the results from the previous analysis and assessment steps to define the V&V strategy that minimizes cost and effort of testing. This can be achieved by doing as much testing as possible on the virtual system model. Indeed, executing tests in a virtual simulation environment is much cheaper and faster than in the physical field. Test planning identifies which methods are to be applied to which system abstraction. This activity can follow approaches adhering to standards, such as ISO/IEC/IEEE 29119.3.

In this task, the user also defines key performance indicators (KPI) that are requirement-defined observable variables of the system. Defining KPIs significantly facilitates specifying what and how to test.

Coverage

Highly automated systems are typically highly-dimensional systems with continuous dynamics interacting with complex and unpredictable environments. Therefore, it is not possible to exhaustively cover this infinite state-space with a finite (and sufficiently small) number of tests. Test coverage addresses this issue by providing a criterion that defines for selecting relevant tests. Coverage metrics enable assessing the quality and completeness of a test suite. Scenarios and KPIs can be used together to identify test goals and hence define coverage criteria that help identifying a small set of relevant concrete tests. Generating test cases from scenario classes often consists in instantiating scenario parameters. Tests can be ordered and prioritized according to their criticality.

Test Model Design and Qualification

The test model is intended to capture the intended properties of the system under test. The main use of test models is to derive test cases from them. Consequently, test models are different from system models. On one hand side, a test model is meant to be an abstraction of the system under test properties that omits aspects, which are not relevant for the testing activity. On the other hand, a test model shall have enough level of detail to accurately represent the tested behaviour.

Qualification is a process to ensure that a test model is sufficiently accurate for its intended purpose. There are several means available to support qualification of test models, including formal verification and model checking methods.

Test Split

This activity consists in clustering the available tests according to the system abstraction on which they will be executed. Test execution in the virtual simulation environment is used to identify which tests need to be performed in a more accurate test environment - XiL (software-in-the-loop, hardware-in-the-loop, etc.) setting, a physical test field or a physical context. The test split is done in ways that will minimize the testing effort but result in sufficient evidence to derive the overall safety argument.

Testing

The actual testing activity is quite similar, regardless of the system abstraction. It consists of four main steps – (1) test case generation, (2) test initialization, (3) test execution and (4) test evaluation.

Test case generation consists of creating concrete test input stimuli and oracles. Usually, concrete tests are generated from the test model that abstracts the SuT behaviour and its interaction with the environment. In the scenario-based V&V methodology, the scenario class takes the role of a test model. The choice of concrete tests that are generated from the model is typically driven by a notion of coverage. Intuitively, a test suite shall cover as much as possible of the model's behaviour. In the case of scenario classes, the notion of coverage can be applied to explore the space of its parameters. However, the question whether a set of scenario classes provides a sufficient coverage of the real world is less clear and more difficult to assess.

In model-based testing, coverage is usually defined with respect to the structure of the model – the test case generation aims at maximizing the number of states, transitions and other structural elements of the model that are explored by the generated test suite. In scenario-based testing, an equivalent definition is the coverage of relevant scenarios in the test process. Relevant scenarios are those scenarios, which might occur in real world and can lead to unsafe, unsecure, unreliable, uncomfortable or incorrect behaviour of the SuT. Statistical model checking (SMC) takes, as its name suggest, a more statistical approach to generating test cases. This method enables sampling the SuT and its environment in a way that provides statistical guarantees about the system with some confidence levels. As such, SMC can play an important role in developing the overall safety argument.

The outcome of the test case generation activity is a finite set of test cases. We note that some test case generation techniques gain knowledge to incrementally generate new tests. Prior to their execution, test cases must be adapted to the abstraction of the SuT – the test case format is not the same for a virtual simulation and a physical test field. Test initialization task prepares a concrete test to its execution in an actual test environment.

Test execution then executes the test suite. In the scenario-based V&V methodology, which promotes virtualizing parts of the system and its environment, co-simulation is an important test execution approach that can be used to perform global *simulation* of a coupled system by composing the simulations of its parts. The co-simulation methods developed in ENABLE-S3 are presented in more details in Chapter 2.4.4..

Executing the test cases results in output sequences generated by the SuT, which are assessed during the test evaluation activity. The test evaluation consists of checking whether the behaviour of the system satisfies functional, safety, security or performance properties. Runtime monitoring is a possible approach for automatically evaluating tests that can be applied during the actual system operation to detect unexpected behaviours.

Testing is a general activity that is repeatedly conducted along several orthogonal dimensions. It is applied across all the system abstractions (virtual, XiL, physical test field and physical context). In the case of testing highly automated cyber-physical systems, testing is typically

repeated under both the perfect and imperfect perception assumption. Testing with perfect perception focuses on the quality and correctness of the control and decision-making procedures, while testing with imperfect perception assesses the impact of inaccuracies in the perception chain to the overall system behaviour. Finally, security testing evaluates the system behaviour under targeted attacks.

We finally mention formal verification that is an optional and complementary activity to testing – due to the underlying complexity of this method and the scalability issues, formal verification can often be successfully applied only to safety-critical parts of the virtual system model. It allows to obtain a proof that the safety-critical component model is behaving according to its specification, thus increasing the confidence in the system correctness that can be ultimately used to help deriving the overall safety argument.

2.1.6 Overall Safety Argument

A safety case is "a structured argument, supported by a body of evidence that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given operating environment" [2]. There are various common argumentation approaches, including conformance to a (non-autonomy) safety standard, proven in use, field testing, simulation, and formal verification. Since technology and safety strategies for autonomous systems are still evolving, it seems likely that safety arguments will be heterogeneous encompassing multiple safety standards as well as various techniques.

All the results of the previous steps need to be collected in a structured way to build the overall safety case, which shows that the final system is acceptably safe.

2.1.7 Application Examples

Application to Use Case "Validation of Valet Parking Function"

In this use case, we approached the validation of an Automated Valet Parking (AVP) system. Even though this operational domain massively reduces the number of different scenarios which need to be tested (compared to autonomous urban driving), purely physical testing is still very costly. Especially since the whole validation process needs to be repeated for modifications of the system implementation. In this use case, we approached the validation of an Automated Valet Parking (AVP) system. Even though this operational domain massively reduces the number of different scenarios which need to be tested (compared to autonomous urban driving), purely physical testing is still very costly. Especially since the whole validation process needs to be repeated for modifications of the system implementation.

The following describes the V&V approach for this use case as well as the embedding of the work in the scenario-based V&V methodology.

Functionality Definition and Operational Design Domain Characterization

As depicted in Figure 2 2, the valet parking system task is to park the vehicles into assigned parking bays or hand them back to their owners at a dedicated pick-up location. When the driver approaches a parking lot with AVP functionality, he or she stops, leaves the car, and activates the AVP system. The Parking Area Management (PAM) locates the free parking slots and defines the path for the vehicle guiding it to the parking lot. The vehicle guidance proceeds under a supervisory control, in which the vehicle and the PAM continuously exchange real-time data and where mitigating operations can be commanded by requiring e.g. emergency braking.

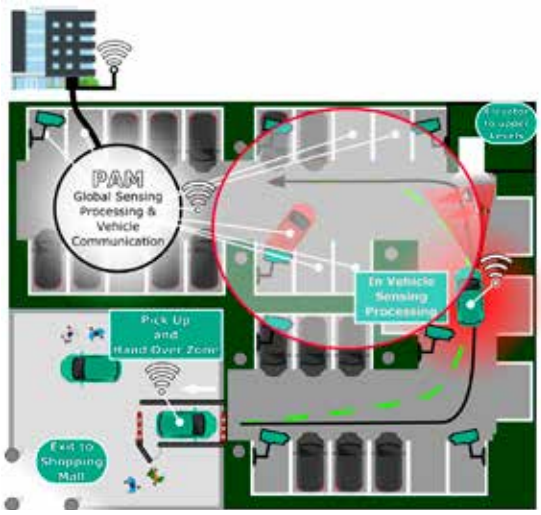


Figure 2 2: Automated Valet Parking System

Scenario Classes Derivation

In this use case, the derivation of scenario classes is not based on real recorded data. Instead, we used a procedure, which virtually generates a diverse set of scenario classes. This procedure was realized in two tools: *OFFIS StreetArt* and *Unimore Map Populator.*

OFFIS StreetArt is a tool for the automated generation of synthetic parking sites. In the first place, these parking sites are tile-based, and thus, very simple yet serve as the basis for the automated generation of synthetic scenarios. OpenDRIVE® was selected as the exchange format for road networks. Figure 2 3 shows the tiles and a randomly generated scenery.

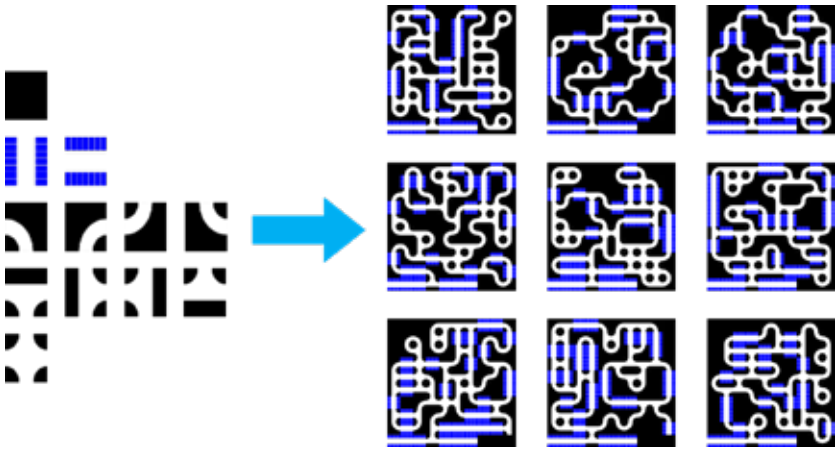


Figure 2 3: A set of bases tiles (left), a randomly generated parking area (right).

Unimore Map Populator is a plugin integrated into OFFIS StreetArt. It fills the maps generated by the latter with objects and actions related to the parking lots: parked cars, moving cars looking for an empty lot or the ones that are exiting their lots and re-enter the traffic again.

The tool uses the parking lots as defined in the map and randomly places vehicles on the map. The vehicles are associated with different behaviours, such as cars entering/exiting their lots (e.g., triggering the exiting car to leave its lot when the tested car is in vicinity) or cars moving around and may make sudden stops to simulate the behaviour of an undecided driver. The Unimore Map Populator permits a user defined ratio of parked cars, ratio of exiting cars, range from the automate vehicles and delay after which a parked car exits, number of moving cars, and number of suddenly stopping cars to obtain scenario classes and instances for this use case.

Hazard analysis and risk assessment

Based on the functionality definition of the AVP and the operational design domain as well as the scenario classes, we performed a hazard analysis and risk assessment as proposed in the ISO 26262. By analysing potential malfunctions of the AVP system regarding their severity,

exposure and controllability we could abstractly classify their safety risk into the Automotive Safety Integrity Levels (ASIL) A to D with ASIL D representing the highest and ASIL A the lowest risk. For each hazardous event a safety goal is derived which inherits the hazard's ASIL. Finally, safety goals are refined into lower-level safety requirements. These can be allocated to architectural components. The validation of each component can be performed according to those safety requirements and a safety concept, as well as a safety-architecture can be developed. This process has been conducted for the AVP system with the outcome of a set of top-level safety goals (shown in Table 1) and the derived safety requirements. These have been applied to the decomposed architecture between vehicle and PAM.

Table 1: List of identified top-level safety goals for an AVP system.

NO.	DESCRIPTION	ASIL
SG 1	The valet parking function shall not be active outside of a PAM managed parking area.	D
SG 2	The system shall prevent collisions between vehicles and persons.	C
SG 4	The system shall not start moving during embarkment and disembarkment.	C
SG 5	The system shall prevent collision with other vehicles.	B
SG 6	In case of a collision or firer the system shall notify a human supervisor.	B
SG 7	The integrity of the communication between the PAM and the Vehicle shall be ensured.	B
SG 8	The system shall ensure that the vehicle stays within the (statically defined) drivable area of the parking area during automated operation.	B
SG 10	The valet parking function shall be disabled when people are inside the vehicle.	A
SG 11	The system shall prevent collision of the automated vehicles with objects.	A

Testing

Having the safety goals and their according safety requirements, test cases can be derived to perform virtual test execution to check the fulfilment of the safety requirements.

Vires Virtual Test Drive (VTD) has been chosen to simulate the environment and the vehicle dynamics. An interface has been developed that allows for data exchange between the virtual simulation and the system under Test (SuT) based on the Robot Operating System (ROS). To

check if the safety requirements are satisfied, they are formalized and compiled to Functional Mock-up Units (FMUs) using the BTC EmbeddedPlatform. These FMUs can be attached as observers to the simulation to check the safety requirements during runtime.

Conclusion

Along the lines of the scenario-based V&V methodology a fully integrated test system for validating the AVP system has been elaborated, which was used to test parking scenarios with multiple automated vehicles coordinated by a parking area manager in synthetic parking environments. The approach and tools have been proven to be capable of testing safety properties of the AVP. Moreover, the high degree of automation in the virtual test system and the diversity of the scenario generator enable the virtual validation of AVP for a large number of scenarios.

Application to use case “Shore based e-Nav-Station with secure data exchange”

Verification and validation of automated navigation functions in the maritime domain is an extremely challenging problem. The inclusion of real ships and its systems such as bridge systems in the testing process is very costly and time-consuming. Sea trials for testing are complex and hardly reproducible. Therefore, V&V activities in the maritime domain cannot rely on physical testing only.

We adopted the scenario-based methodology using co-simulation to overcome this problem. We used requirement and scenario elicitation, combined with scenario filtering, to come up with representative scenarios that we applied in a virtual co-simulation environment.

The main goal in this use case was to use the scenario-based V&V methodology to assess systems that support the ship crew with automated navigation and remote vessel guidance at the high sea. Such systems perform decision making more automated and thus increase the resting period for the crew, and consequently reduce human errors due to fatigue and lack of concentration.

For the elicitation of operational scenarios and scenario classes, we started by analysing collision databases and creating scenarios from the real-world recordings. We analysed

- hundreds of normal traffic scenarios,
- over 200 collisions (cf. the plot in Figure 2 4),
- 120 groundings,
- accidents with misunderstanding of COLREGs (maritime traffic rules), and
- Real Operation Context of sea traffic (operational design domain).

Based on the above, we systematically mined scenario classes that describe in detail the initial conditions for the configuration of ships/agents, environmental, and operational aspects. For instance, this includes the type and placement of agents in the geographic environment, their physical and operational capabilities, and their planned trajectories; specifics about environmental conditions, including weather, daylight, water currents and background noise, such as the number and distribution of neutral entities like any unrelated marine traffic. Such a scenario class may involve dozens or even hundreds of agents and complex conditions.

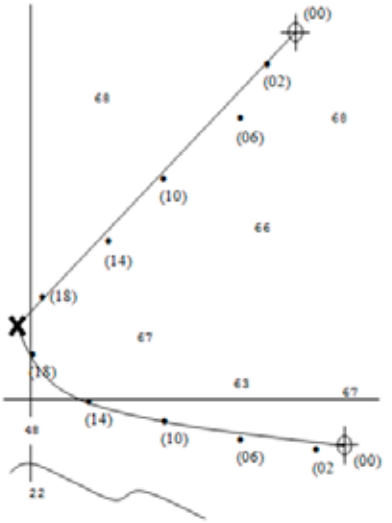


Figure 2 4: Plot and Detailed Time Series of the Trajectories of Two Ships Ending in a Collision

In total, we manually derived over 60 scenarios and deployed them in the virtual co-simulation setting modelling the ship and its complex environment. These scenarios have subsequently been used to test a navigation component. This use case is probably the first extensive scenario-based V&V of an automated navigation function in the maritime domain.

2.2 Big Data Analysis and Scenario Detection

Key Results:

- A shared language and approach for using scenarios in safety validation, including a definition of scenario class and relevance to be used within the project. Before the project there have been several proposals for terminology for the various application domains.
- Systematic overview of available data sets from the ENABLE-S3 project or other public sources
- Development of scenario detection algorithms for activity or manoeuvre detection, scenario detection and distinguishing driver and automation.

- Algorithms for critical case identification, test case generation and OpenDRIVE®/ OpenSCENARIO® generation

- Tool integration, including integration of StreetWise scenario database with test case generator and simulator.

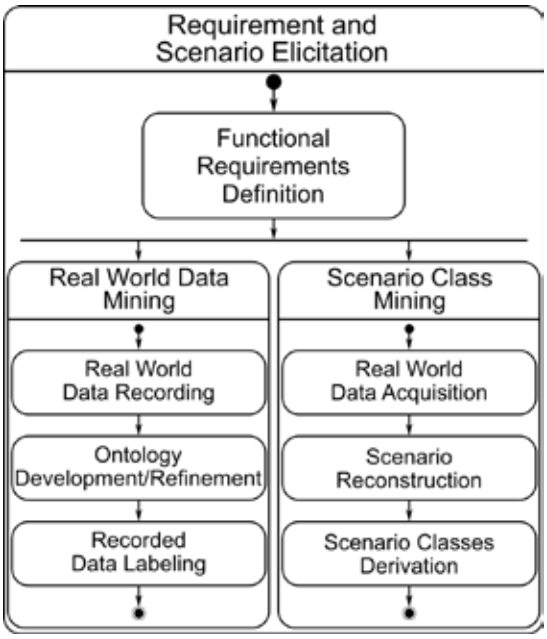


Figure 2 5: Requirement and scenario elicitation

In automated cyber-physical systems, the role of the operator (e.g. driver in a vehicle or tractor, pilot in an airplane, captain in a ship) is transferred partially or fully to the system. This means that the automation system has to successfully deal with 'all possible situations' that may occur in the real environment. Given the highly multidimensional and variable character of these possible situations (or 'scenarios'), it becomes impossible to define a set of test cases that cover all of them.

Therefore, a structural means of collecting scenarios is needed. Since automated systems are equipped with sensors and logging, a rich source of information is available. This chapter summarizes the developed methods and techniques to analyse big data to detect scenarios. Scenarios have the advantage of reusability across a wide range of system variants and generations.

This chapter on big data analysis and scenario detection follows the structure shown in Figure 2 5 and described in Section 2.1.2 from the advocated overall V&V methodology.

Requirements and Scenario elicitation

In systems engineering, there is a long tradition of collecting functional requirements about what the system should do for the end user. This can be done through reuse of requirements of previous system generations, interviews with users and clients and through translation of service calls and incidents to new requirements. These requirements are then translated into technical specifications.

Functional requirements definition

In ENABLE-S3, an additional source of functional requirements is provided by scenarios. They can be generated by engineering or based on a model, observed through data collection or resulting from a safety / security analysis. For each scenario, the desired or intended behaviour of the system should be defined. This will create a rich source of requirements.

2.2.1 Real-world data mining

Given the fact that modern systems have sensors and logging built in, an enormous source has become available for information about system use and behaviour of operators and other stakeholders. A frequent or continuous process can be set up for mining such data. This holds the promise of testing a system against scenarios collected during real-world use of the system.

Several organizational challenges exist for data collection [FOT-net 2019]:

- data protection
- privacy of personal data
- ownership of data (user or system owner or manufacturer)
- transmission and storage of huge data sets
- maintenance cost of database

Real-world data recording

The real-world data recording looks around the system (‘environment sensing’), may include monitoring of operators and looks inside the system (‘system state sensing’).

Environment sensing: the quality and field of view of the sensor set will define the quality of the data collection. If a car has no sideward sensor view, overtaking cars are detected late. Requirements to data sets have been formulated.

Monitoring of operators: the operator activities and position remain essential, even in highly automated systems, as the interaction between operator and automation is often safety-critical. For example, if the position of the surgeon is not measured with a sensor, it cannot be included in the analysis of usage patterns of an interventional X-ray machine in surgery. In some cases, it is sufficient to observe the external behaviour of operator and system together.

System state sensing: essential information about the system under test are position on global and local level, speed and direction. Also, the information which automated functions are activated is essential.

There are many decisions on what to measure or not. The data collection engineer needs to make a trade-off between using production systems with standard sensors that can produce much data quickly or special measurement systems with elaborated sensor sets that can give high quality insights at high costs.

Logging of internal software events is not limited by sensors but primarily by processing power and storage space. Also, it is needed to determine the ground truth ('what really happened?'). For this, a simple camera with manual annotation or a more elaborate reference sensor set can be used.

For scenario class mining, typically the raw signals are processed from pixels to tracked objects by sensor fusion (or 'world modelling'), e.g. a particular ship following a certain trajectory. This reduces the data size considerably. For later use of the data set, proper documentation of the data set is essential. As part of an overview of data sets, a format was defined for data set documentation.

Ontology Development/Refinement

For sharing scenarios, storage in shared scenario databases and for making safety evidence comparable, a common language on scenarios, scenario classes and their elements is essential. Through real-world data recording, we collect a view on the world, but how to name it? The terms used should be comprehensive, not overlapping, commonly used and coherent. For this, an ontology is very appropriate. This is a limited set of terms that describe the categories that we observe. This should describe the dynamic environment, the static environment and environmental conditions.

Key Result: Scenario Class Definition

As ENABLE-S3 has many partners, many different versions of terminology and dialects existed. Therefore the ENABLE-S3 scenario working group therefore started with definitions of the term 'scenario' and its components. It turned out that scenarios can be grouped in scenario classes that make handling them much easier.

After reviewing many existing definitions, ENABLE-S3 formulated the definitions used throughout the project (see also chapter 1.1.1):

A scenario class is a formalized description of the multi-actor process, including its static environment, its dynamic environment and environmental conditions.

In a scenario class, the parameters are described and may have parameter ranges or distributions.

A scenario class may include activities, events, goals of the activity and decisions of actors.

Other important definitions are listed in Table 2:

Table 2: Scenario components and definitions of terms

SCENARIO COMPONENT	DESCRIPTION
Scene	A scene describes a snapshot of the environment including the scenery and dynamic elements, as well as all actors' and observers' self-representations, and the relationships among those entities.
Activity	An activity refers to the behaviour of a particular mode of a system.
Event	An event marks the time instant at which a transition of state occurs, such that before and after an event, the state corresponds to two different activities.
Situation	A situation is the entirety of circumstances, which are to be considered for the selection of an appropriate behaviour pattern at a specific point in time. It entails all relevant conditions, options and determinants for behaviour.
Test case	A scenario selected from a broader scenario class with all values specified to be used for testing within the operational domain of a function under test.

Recorded Data Labelling

To make large data sets manageable, labelling typical sections of the data is very useful. In this way, the data set can be viewed as a storyline and can be searched for certain types of activities. Also, base statistics per label can be given ('21 pedestrians per hour').

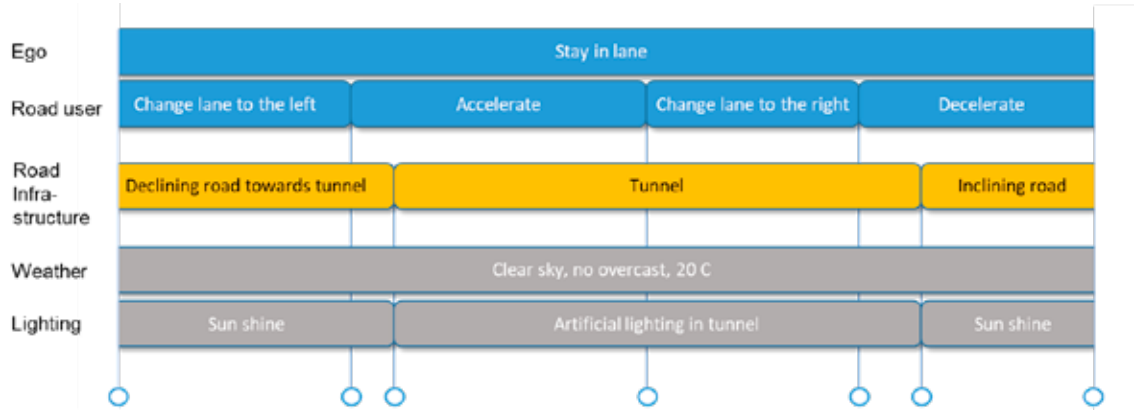


Figure 2 6: Storyline based on activity labelling (TNO, 2018)

Together, the labels used also form an ontology as described above. They can be defined by desktop research, by consensus or by observation. The ENABLE-S3 project has shown that labels should be verified or defined by observation as real-world scenarios are typically much richer and more complex than we can imagine.

Labelling can be done manually by humans by watching and annotating recorded video sequences. However, the various people doing this tedious work should all use the same interpretation of the label. Therefore, independent labelling by two or more people can be done for the same data to verify the consistency.

To avoid costly manual annotation, automated labelling is very desirable. Using algorithms, predefined patterns, activities or events can be detected. To validate the quality of labelling algorithms, they need to be compared with manual annotations on a large enough validation data set ('the ground truth'). After that, they can be used routinely. TNO developed activity detection algorithms focusing on single road user activities. It is also possible to directly label at the level of scenario classes (see next chapter).

Key result: tool chain for ground truth generation from real-world data

The Hella Aglaia Ground Truth Generation system (GTGEN, Figure 2 7) provides a complete workflow for annotation of real-world data. Using automated and manual annotation in a web-based tool, ground-truth labels are created.

The following label files can be generated for each recorded video file:

- automatically generated labels of the dynamic environment like vehicles and pedestrians (including object classification, bounding box and trajectory description)
- manually generated labels of the static environment like street and traffic sign description (AnnoStation web tool)
- automatically generated labels related to the ego vehicle motion and extracted from the recorded vehicle bus data

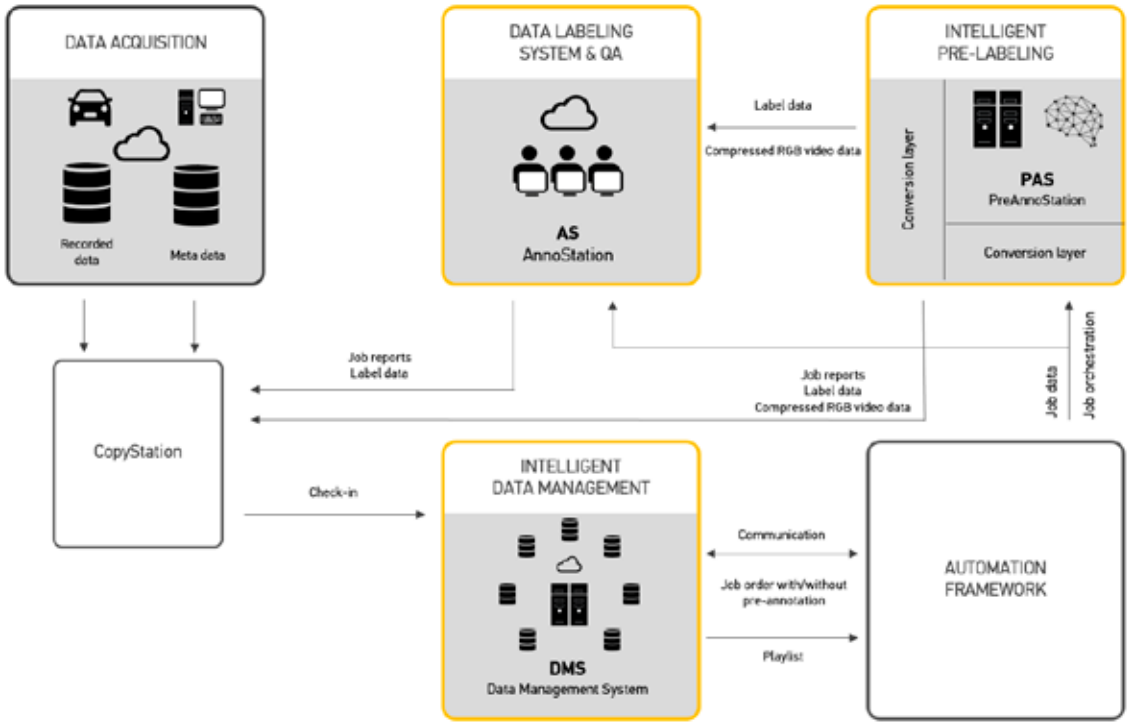


Figure 2 7: Tool chain for Ground Truth Generation (Hella Aglaia)

The tool for ground truth generation was used in the automated valet parking use case, including the automated processing workflow. For test purposes it is necessary to include critical test cases, which are very difficult or not possible to catch during data collection in normal real traffic environments. Such critical test cases can be generated in the simulation using a non-critical real-world scenario by varying the parameters describing the ego vehicle motion and/or the motion of the other traffic participants. For this an OpenDRIVE® / OpenSCENARIO®

file is generated using the results of a combined search in the data management system for a wanted scenario. In this way, GTGEN supported the evaluation of the representativeness of generated reference scenarios with real-world data for valet parking.

Key result: elements of representativeness

When collecting real-world data, quickly the question arises: have we collected enough data? Does the collected data now adequately represent the real-world use of our automated system?

Not surprisingly, this really depends on what one wants to know. In order to learn more about the average speed on a German highway, one can collect data of diverse highways, covering all times of the day, week and year and various weather conditions. However, if one wants to characterize the European driving behaviour including variations of speeds, distances, etc. the task becomes much more elaborate.

Once the collected data does not show “anything new”, the collected scenarios become representative. It does not imply that every possible scenario has been observed, but that we have captured all the variations and probability of the variants.

By comparing distributions of parameters for about 90% of the data and 100% of the data, we can see whether we have learned anything new on this parameter in the last 10%. If not, our data is considered representative for this parameter. If several parameters are dependent, the amount of data required increases. To reach representativeness, the following elements are important:

- Regional spread of data collection.
- Time of day, week and year
- Types of road, terrain, sea or other system environment
- Weather, lighting, dust variants
- Quality of the sensor set, and width of the system signals available. In particular field of view: if a scenario starts out of view, a complete description is not possible or needs to be estimated.
- Quality of data labelling algorithms. The algorithm will not always be perfect. Metrics exist for this.
- Quality of scenario reconstruction algorithms. As above.
- Quality of annotation. This is used to measure the quality of the algorithms above. Metrics are available.
- Quality of new pattern or scenario detection. Some patterns or scenarios may not have been observed or defined before. Can the algorithms detect such new phenomena?
- Quality of parametrization. How much information did we loose by using a limited set of parameters? Metrics exist for this.

2.2.2 Scenario Class Mining

Scenario class mining (or 'scenario detection') automatically derives scenarios from a data set. Scenario class mining consists of:

- Real-world data acquisition
- Scenario reconstruction
- Scenario class derivation

Real-world Data Acquisition

Scenario class mining can be done using previously collected data sets, public data sets or fresh data. This has the advantage that scenario detection can be done off-line while new data collection is going on in parallel.

Key result: overview of ENABLE-S3 and public data sets

ENABLE-S3 created an overview of data sets available within and outside the project, mostly observed and sometimes created synthetically from a model. This includes requirements to data sets, a data set documentation format and an initial .json meta data format for data sets.

Public data sets suitable for benchmarking are:

- Vision: Karlsruhe KITTI, Berkeley DeepDrive, CityScapes, nuTonomy NuScenes, Mapillary Vistas, ApolloScape
- Radar, lidar: nuTonomy NuScenes
- Scenario detection: TNO Benchmark data set for scenario detection

Scenario Reconstruction

Based on predefined characteristics of a scenario, the various raw signals, tracked objects or labels are used to derive the scenarios observed. This can be done in many ways:

Supervised Machine Learning Techniques

- Template matching algorithms such as dynamic time warping or rule-based algorithms (which sections of the data look close enough to this manoeuvre, shape or pattern?).
- Mixed models based on physics and engineering know-how: using system dynamics models or other engineering knowledge of the system, the data is transformed to a higher level of information. Then other techniques are applied.
- Machine learning techniques that are trained on the data together with manually labelled

correct scenarios (neural networks, generative adversarial networks, decision trees, Markov models, support vector machines, etc.). After training, these models derive the scenarios based on the underlying labels or raw data.

Unsupervised Machine Learning Techniques

- Clustering and label proximity techniques such as hierarchical clustering, K-means, N-Grams (which activities happen a lot in combination?)
- Model creation techniques for latent variation such as principal components analysis, independent component analysis, singular value decomposition, Bayesian models, Markov chain (What patterns exist in the data? What dominant dimensions are in the data? How can I represent the data with a limited set of parameters?)
- Black box techniques like autoencoders. These techniques do lead to effective predictions but do not give additional insight why. The outcome can be unpredictable if used outside the training domain.

The supervised techniques assume previous knowledge about a scenario (shape matching, mixed models, trained machine learning techniques). This means you will not find unexpected scenarios. Unsupervised techniques look for combinations or patterns without previously defining a scenario (label proximity, model creation techniques). This has the advantage that scenarios are detected that 'you did not think of'. The disadvantage is that they might find patterns in the data that do not fit to the human notion of scenarios, and hence do not make sense to humans.

The method used for scenario reconstruction also determines its scalability: It analysis it is necessary to create a new algorithm for only every scenario class, or if it possible to create algorithms, whcich can detect several scenario classes? Here the unsupervised methods have an advantage. An example of unsupervised machine learning is to distinguish between the operator actions and the system automation. Usually, from the outside it is not clear whether the operator or the automation takes a decision. IBM and UCD elaborated this:

Key result: Critical case generation based on parametrized real-world driving

JKU developed a catalogue of scenario classes, consisting of 22 scenarios. It has been built up through analysis of the crash database of The Second Strategic Highway Research Program (SHRP 2) Naturalistic Driving Study (NDS) and is proven to have a coverage up to 99% of critical (leading to collisions) events on highways recorded in SHRP2 NDS.

A data-driven method has also been proposed to achieve the parameterization of each scenario class based on real world measurements (Figure 2 8). Through variation of parameters'

values within their range, we obtain various specified scenarios for safety testing of ADAS, which covers up to 90% of the corresponding real-world measurements.

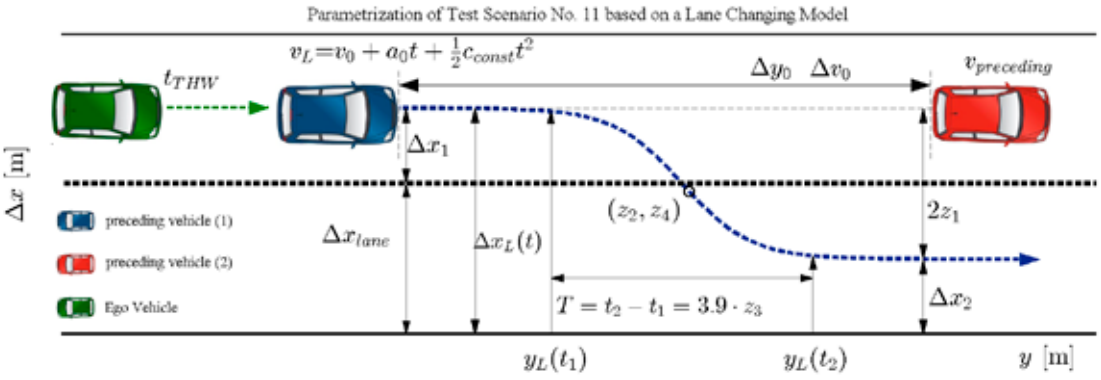


Figure 2 8: The exemplary parameterization of a scenario class

For parametrization of several of the scenario classes, templates were developed that characterise the manoeuvre of each road user in such a scenario class. Using elastic template matching, the template is stretched in such a way that it optimally fits the data in a sliding window. The fit is optimised using the dynamic time warping cost.

The same technique was used for detecting new scenario classes based on lateral distance between vehicles. To detect new scenario classes, k-means clustering is applied to the observed scenarios after template matching. In this way, on top of the initial cut-in scenario, pull-out, following and in-between lane driving were detected.

For above parameterized scenario classes, an Input-Design method has been developed. It searches in an efficient way for a safety boundary, separating safe conditions from crashes. This boundary represents the limitation on performance safety of the ADAS with respect to real traffic (corresponding to measurements), and accordingly the region of interest in terms of safety (Figure 2 9).

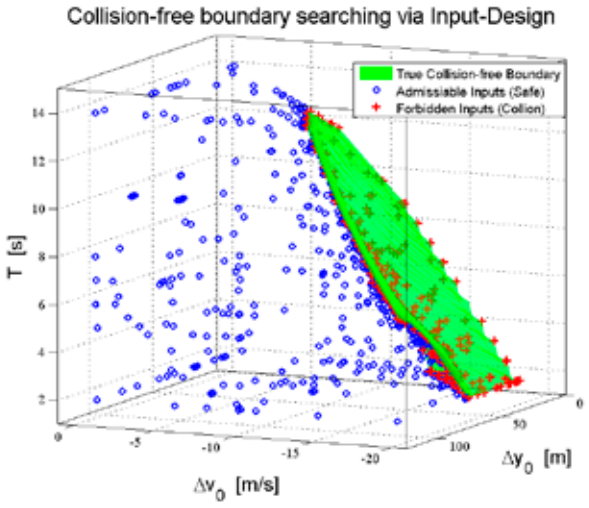


Figure 2 9: The resulting safety boundary of the simplified 3-dimensional case from Figure 2 10

Scenario class derivation

Scenario reconstruction may lead to a high number of scenarios that may or may not be meaningful to a human. Also, they may be very similar in some respects. For high reusability, categorizing scenarios into scenario classes is very useful. The scenario classes make use of the similarity of the underlying scenarios to provide a much more condensed list. Typically, we talk about a 'cut-in scenario', the scenario class with all of its variants. Describing a data set with scenario classes has the similar advantage as labelling sections of a data set: it makes it more manageable and reusable. With scenario classes, the description is at a higher, more abstract level.

From engineering experience, a number of 'obvious' scenario classes may be quickly found ('X-ray sensor close to patient'). After that, very soon the question pops up whether a new scenario is a variant of a previous scenario class or a new class in itself ('split or merge'). This is a classical discussion for many people who use scenarios. There is no scientific or objective answer to this question, as a hierarchy or scenario classes at arbitrary level of granularity can be made. Therefore, the list of scenario classes can only be made in consensus. The ISO working group ISO/TC 22/SC 33/WG 9 'Test scenario of autonomous driving vehicle' will eventually define such scenario classes.

Scenario classes do not include the component or system that is tested, nor a metric for evaluation (KPI). Scenario classes are reusable across functions and across metrics, which makes them all the more valuable.

A test plan will combine a function under test, a scenario, metrics and pass/fail criteria. Naturally, it is possible to make a table with scenario classes that need to be covered at least for testing a certain function. However, other scenario classes may also turn out to be relevant.

Key result: definition of scenario relevance

Now that we have collected all these scenarios and defined scenario classes, which are the important ones? This depends on what you want to test. ENABLE-S3 created a definition of relevance of scenario classes:

Relevant scenarios are those scenarios classes that help discriminate between intended and undesirable behaviour.

To our knowledge no definition of scenarios' relevance for complex or automated systems have existed in literature so far. Intended behaviour can relate to safety, comfort, reliability, security, fuel consumption or other areas. Typically, intended behaviour is captured in requirements.

Parametrization

It is important to capture the variation within a scenario class so that in testing, the real-world probability of the test set can be known. Since a scenario class contains many individual observed scenarios, the variation within a scenario class can be described. This can be done on the level of the raw signals, the individual road user activities (e.g. trajectories) or on the level of interactions between multiple road users (e.g. takeover).

Most commonly this is done by parametrization: describing the variation with a small number of key parameters, e.g. lateral speed, relative longitudinal distance, etc. To describe the variation, a balance needs to be found between simplicity of the description (number of parameters), leading to strong data reduction versus the accuracy of the description, leading to low loss of information.

Tool integration

Most simulation tools can handle scenarios defined in a manual or automated way. The road environment can be imported from OpenDRIVE® by many simulation tools. Vires VTD and Siemens SimCenter PreScan can read OpenSCENARIO® files that specify the dynamic part of a test case.

For running the simulations, vehicle, driver and sensor models can be specified in the tools or imported as FMU (black box container for models) in several tools. ENABLE-S3 initiated a new real-time Open Simulation Interface (OSI, see section 2.4.2).

Key result: Integration of TNO StreetWise scenario database with AVL Test case generator, AVL Model.CONNECT and VTD simulator

TNO developed the StreetWise scenario mining pipeline and database [Elrofai 2018] as a MS Azure cloud solution. This pipeline (Figure 2 10) covers the following first 6 steps (orange and blue):

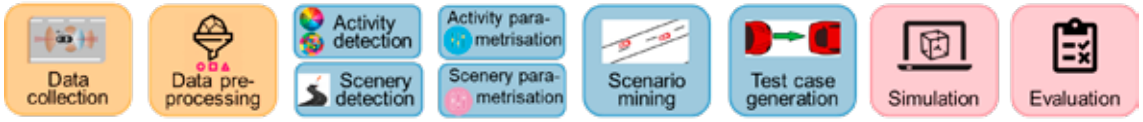


Figure 2 10: TNO StreetWise pipeline

The scenario database provides statistics on exposure to scenario classes and percentile covered by selected parameter ranges. It generates OpenSCENARIO® files based on the scenario database following the test specifications entered.

For its verification and validation tooling, AVL developed a Test Case Generator (Figure 2 9) which specifies the desired tests through region, road type, scenario classes, parameter ranges and number of test cases. Through the StreetWise API, it collects the resulting OpenDRIVE® and OpenSCENARIO® files from TNO StreetWise. Through AVL Model.CONNECT, the tests are automatically configured and run in Vires VTD. In this way, the full process from test plan to simulation runs is integrated and can be automated.

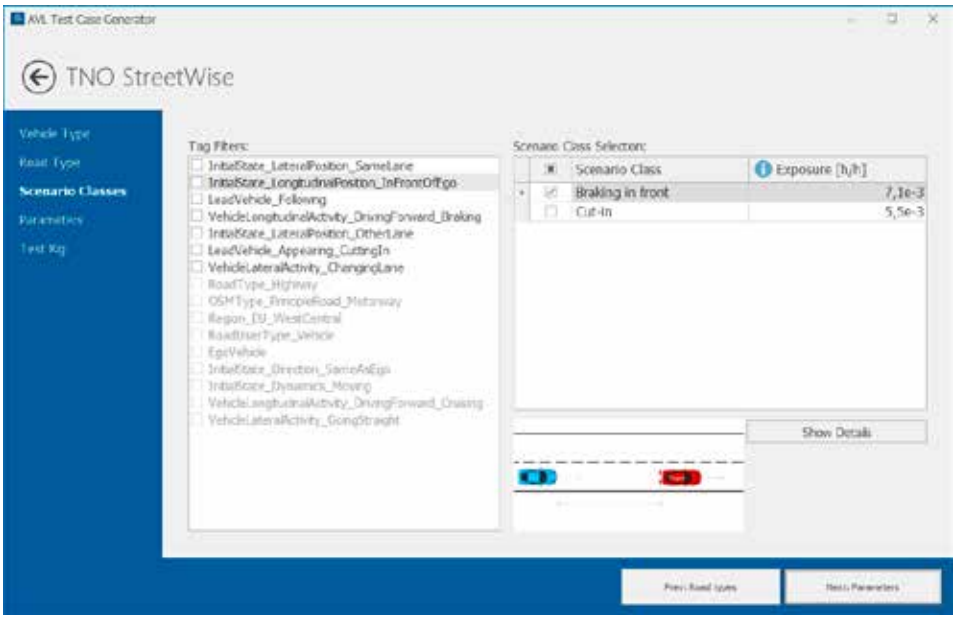


Figure 2 11: AVL Test case generator reading from TNO StreetWise database

Further reading

- FOT-net 2019, Data sharing framework 1.1, result of EU FOT-net and CARTRE projects. See also http://fot-net.eu
- ENABLE-S3 deliverable "D41 (D3.5) D3.1.5 List of recorded Datasets", to be published on the official project website https://www.ENABLE-S3.eu/media/deliverables/
- H. Elrofai, J-P Paardekooper, E. de Gelder, S. Kalisvaart, O. Op den Camp, StreetWise Scenario-Based Safety Validation of Connected and Automated Driving, white paper, 2018, retrieved at http://publications.tno.nl/publication/34626550/AyT8Zc/TNO-2018-streetwise.pdf
- Epperlein, J.P., Zhuk, S, and Shorten, R. "Recovering Markov models from closed-loop data." Automatica 103 (2019): 116-125.
- Leitner, A., Watzenig, D. (editors), "Validation & Verification of Automated Systems – Results of the ENABLE-S3 Project", 2019, Springer Verlag (to be published)
- Zhou, J., Del Re, L. Identification of critical cases of ADAS safety by FOT based parameterization of a catalogue, 2017 11th Asian Control Conference (ASCC)

2.3 Validation and verification tool development

Key Results:

- Implementation of tool support for the developed methodology.
- Gaps in tool chains have been closed in order to have a seamless (and thus more efficient) test environment.
- 9 new tools have been developed
- 27 tools have been enhanced with new functionality developments and have been put to market
- 57 new tools or tool extensions identified

The goal of WP3.3 "Validation and verification tool development" was to fill gaps in tool support to implement the scenario-based validation and verification approach of safety and security aspects of ACPS to increase developer productivity, and thus, allow safer and more secure ACPS systems to be developed in a cost-efficient manner. The work has concentrated on enhancing capabilities of existing tools and adapting existing tools to better fulfil the safety and security validation and verification requirements of the use cases. In addition, new tools have been developed if existing tools have been lacking validation or verification capabilities.

Another important feature for tool development in ENABLE-S3 was the fact that the use cases are from six industrial domains, and the state of the art of safety and security validation and verification between the domains varies substantially. For example, the use of advanced simulation tools and closed-loop testing is part of the established V&V practice in some domains (e.g. automotive), while in some other domains only first steps toward that direction had been taken at the beginning of ENABLE-S3 (e.g. maritime). ENABLE-S3 has facilitated the alignment of V&V practices by developing cross-domain tools and eased the transfer of best practices identified among domains.

The development of the tools has been done in close co-operation with the industrial use cases, but in order to produce critical mass in tool development, the tool development has been focused on selected tool areas. The tool areas are as follows:

- Automated test design
- Simulation based testing
- Automated validation and verification
- Validation and verification process support

Most of the use cases use tools from several tool areas. A common approach in the use cases is that high level models, often defined using a domain specific language to make them more comprehensible for the domain experts and end users, are used in very early phases of the development process to validate system properties. For example, the model can be used to simulate the system under development so that users may experiment with the system and assess whether it works as expected. In later phases the model is refined to a level allowing formal analysis of system properties. Finally, the model is used in scenario and test case generation in the verification of the implementation, in coverage analysis, and in the estimation of the remaining safety and security risk. This kind of tool interoperability is possible only if the tools support common formats and artefacts. A lot of the development work has focused on tool interoperability.

Closed loop simulation is used heavily in the ENABLE-S3 use cases. The environment of the system under development and its interaction with the environment are simulated so that the system can be tested. Tools in the use cases support model-in-the-loop, software-in-the-loop, hardware-in-the-loop, and system-in-the-loop testing. As the use cases are from six different domains, a lot of domain specific extensions have been developed, e.g., environment simulation for farming domain.

Below we describe the tool areas in more detail and give summary of common themes in tool development.

2.3.1 Automated test design

Quality assurance, especially safety and security validation and verification, of autonomous or highly automated cyber-physical systems is even harder than quality assurance of ordinary software. This difficulty lies within the open world nature of the operational environment of cyber-physical systems. “Open world” means that only the laws of physics restrict to what happens in the physical environment of the system. In this sense, the world around the system is open. The system survives, or performs at least as well as a system operated by a human being, in every scenario it encounters and we should be able to verify this.

Tools in this area are aimed for helping in designing how the system should be tested. In particular, tools filling gaps in automation of the following tasks have been developed:

Automated scenario generation, test case instantiation, and extraction from measured real-world data.

- Automatic generation of safety and security test cases. New test case generation tools are implemented, and existing tools adapted and optimized to consider safety and security V&V of ACPS. These tools are based on methods including model-based and muta-

tion-based test case generation, fuzzing, and probabilistic models. Test cases are generated for, e.g., camera-based sensing and perception.

- Automatic design of test suites. Tools for test case selection are developed to accelerate test execution by avoiding redundant test data and selecting those test cases that unveil yet unknown failures. Moreover, efficient coverage metrics are implemented to aid evaluation of test coverage of generated test suites. For instance, coverage metrics for visual aspects of camera-based sensing and perception, as well as, statistical test-coverage are considered.

- Variability of ACPS systems and components. Tools are implemented and adapted to minimize effort of validating test cases for high number of different models and types of ACPS caused by variation and modifications.

The tools in this area are related to the two upper layers of the ENABLE-S3 generic test architecture, i.e., *Test Data Management and Test Management layers.*

2.3.2 Simulation based testing

The state of the practice on using simulation-based safety and security validation and verification varies a lot between the domains. Therefore, the work in this area very much concentrates on adapting tools to domains that were lagging behind and filling gaps. There are two main threads:

- Virtual product testing aims at facilitating early validation of the system under development. This is achieved by modelling and simulating the system and its components, e.g. by providing system component models to simulate and analyze different system configurations. The goal is to help developers to do it right from the very beginning through early validation of system properties and thus to drastically reduce the development effort.

- Closed-loop testing. This includes development of elements related to the simulation of the environment in which the system under test operates. For instance, environmental models addressing weather and environmental disturbance effects, such as rain and dust, were developed for algorithms relying on video and radar data.

The tools in this area are related to all layers of the ENABLE-S3 generic test architecture, i.e., *Test Data Management and Test Management layers*, and *Test Execution Platform.*

2.3.3 Automated validation and verification

Tools in this area aim at automating safety and security validation and verification. Highly automated analyses reduce costly human work and chance of human errors. These tools use artefacts from different phases of the software development process, e.g., requirements, design, and source code, and check their conformance to specified safety and security criteria.

Used techniques include, e.g., formal methods, model-checking, static analysis, and dynamic analysis. These are used, for instance, for formal verification of safety requirements, verification of information flow, and detection of potential vulnerabilities in program's source code according to vulnerability patterns.

The tools are related to the two upper layers of the ENABLE-S3 generic test architecture, i.e., *Test Data Management* and *Test Management*.

2.3.4 Validation and verification process support

Verification and validation must be performed throughout the whole software development life-cycle and significant part of development time and cost is spent in verification and validation activities. Automated tools reduce costly human work, but they must be tightly integrated in the development process in order to be used efficiently. Furthermore, different V&V activities and tools do not function in isolation but require artefacts from different development phases. Main themes in this task include:

- Development of tools to ensure and improve traceability between safety and security requirements and the validation and verification process.

- Integration of verification and validation tools as part of software development processes including, e.g., testing work flows, as well as, product line development and continuous integration approaches

The tools are related to the two upper layers of the ENABLE-S3 generic test architecture, i.e., *Test data management* and *Test management*.

2.3.5 Summary of common themes in tool development

In testing cyber-physical systems, it is not enough to ensure that the internal coverage of the test cases is high, i.e., the test cases cover all or almost all parts of the system's internal structure or functioning. Software level statement or branch coverages are examples of this kind of internal coverage metrics. Internal coverage can be measured for models of the system, too. The basic idea in internal coverage is to measure how extensively a system's internal structure has been tested. For highly automated or autonomous cyber-physical systems this is not enough as we need to ensure that the system functions as intended in every scenario. In validation and verification of automated cyber-physical systems it is essential that, with high likelihood, for every scenario the system encounters in actual use, there is an "equivalent" test case. Equivalent means that the scenario is similar enough so that we are confident that if the system passes the "equivalent" test case, then the system functions as intended in the actual situation. The notion of a scenario captures this idea. A scenario class can be parametrized. The parameters characterize various aspects of the scenario class, e.g., weather conditions, environment, and behaviour of other objects. Several test cases can be derived from one scenario. External coverage can be increased by carefully generating, selecting, and instantiating scenarios and then generating the actual test cases from the scenarios.

Several tools have been developed to ease scenario-based virtual validation and verification. These tools support:

- extract scenarios from real-world data recordings
- generating synthetic scenarios algorithmically
- combining complex scenarios from simpler ones
- improve scenario instantiation with respect to external coverage
- generation of test cases from scenarios

Another common theme is to fill gaps between model-based testing tools. Model-based testing tools to automatically create test cases and oracles for these test cases were developed further. In this context model-based techniques provide several advantages. Firstly, the size and coverage of the test suite can be optimized. Secondly, the same models can be utilized in other ways in the development process: It can be verified that the required safety and security properties hold in the model. The models are also used in simulation. E.g., scenarios are simulated in a model-in-the-loop setting. This allows validation of correctness of the model's behaviour and early detection of errors. The models are also used in assessment of test suite's capability to detect errors. Tools in this area utilize model mutation. The tools can generate test cases for mutations that are not detected by the test suite, and this way improve test suite's error detection capabilities.

The farming and maritime use cases are good examples of simulation and co-simulation tool adaptations to new domains. Existing tools have been extended to support simulation of new domain-specific environments and components. The benefits of simulation-based validation and verification in these domains are immediate. In both domains it is expensive to test in real environments. E.g., when testing harvesting functions of a harvester, a field is needed. If the test cases must be repeated, then, in the worst case, you have to wait for the next crop and proper environment conditions. Physical testing with real vessels in the maritime domain is extremely costly and risky.

The boundary of the system under test is critical in validation and verification of safety and security aspects of the system. To allow extending the boundaries of the system under test in closed loop testing, in particular in hardware-in-the-loop testing, simulation tools were further developed to allow simulation of sensor stimuli. This allow the use of the real product sensors – that is the actual perception chain - in simulation instead of simulating the output of the sensors. In this context it is essential that high fidelity sensor stimuli can be generated for various environmental conditions (e.g., weather).

The use cases show that in practice the tool areas are related to each other. As mentioned earlier model-based techniques can be used, and are used, in (1) test design, (2) in simulation-based validation and verification, and (3) automated safety and security validation and verification. In many use cases all these areas are involved and gaps between tools are filled so that tools can be used together, and they can use same artefacts.

It was observed that organizational aspects are important as they affect acceptance of new methods and tools. E.g., use of model-based techniques provides many benefits but their use requires special expertise which is often missing from domain experts. Another undesirable phenomenon is that the models must be up-to-date. This is exacerbated by the fact that there are often models on several abstraction layer. This challenge can be mitigated to some extent by using domain specific languages. In fact, it has been proven to be useful specifying a user interface design using domain specific languages. The key idea is that the specification of the user interface design when defined in a formal model-based way, using domain specific languages, can be used for generating multiple artefacts. Firstly, a formal specification should eliminate miscommunication introduced by ambiguity in the definition of the system specification. By specifying the user interface design using a domain specific formal language several specification documents can be generated from it in different abstraction layers.

Moreover, the user interface design can be used for specifying new system configurations used in virtual prototyping to facilitate communication between stakeholders, for system code generation, as well as for model-based testing by generating the models for different model-based testing tools (decoupling the models from tools). The advantages of using user interface design specified in domain specific language and thereby maintaining one model as

single source of truth are manifold. Firstly, the burden of maintenance of different models is reduced, by maintaining only one central model. Secondly, the different abstraction layers of the models enable the use of the user interface design by different stakeholders. Thirdly, the use of a non-proprietary model definition toolkit enables the decoupling of the current model specification in proprietary tools, while opening an opportunity to generate artefacts for all kind of different domains in the model-based system engineering field. Fourthly, the model-based testing models are generated at the early specification phase of the V-Model enabling a rather fast system verification that can be used during the development phase. Fifthly, the user interface design specified using domain specific languages is usually better accepted by different stake holders, thereby enabling fast and easy adoption by the organization.

2.4 Simulation platform for system validation

Key Results:

- We have integrated function simulation with environment simulation. Before ENABLE-S3 mainly functional co-simulation concepts existed that had to be extended since they did not fulfill the requirements for scenario-based simulation. Now, it is possible to simulate automated system functionality in different traffic scenarios under various environmental conditions.

- We have applied and advanced real-time simulation which allows to test functionality with real hardware in simulation-based test frameworks. Within the project we developed methods and advanced tools to improve the performance of co-simulations. Now, it is possible to fulfill real-time requirements which enables the integration of hardware into the simulation.

- We verified autonomous system functions in distributed co-simulation environments. In order to cope with the challenge of exchanging large data sets needed for environment data, we extended existing methods and tools. Now, we have demonstrator setups working across companies to show distributed simulations via a large distance.

- We have aligned methods between different domains. We worked together across domains to exchange experiences and understand each other's challenges. This helped us to improve existing methods and tools that can now be used cross-domain.

  We have cooperated with several other projects to establish standards for simulation-based

testing of highly automated systems. Further, we participate in standardization groups to bring in our experiences and recommendations. Now, specifications exist to help sustaining our project results.

In the context of the generic test architecture, this work package focuses on the third layer '*Test Execution Platform*' (see Figure 1 4). This layer describes the main components of the test system that are necessary to virtually test autonomous functions based on scenarios.

The integration platform enables the coupling of various vehicle, ship, railway, or airplane automation functions (e.g. controller, driving function, vehicle dynamics) to test components and whole systems. The main idea is to have components (simulation models, software or also hardware) that are developed individually. These components receive input data required for computations and provide computation results to other components. By coupling various components in a closed-loop simulation, a more holistic system simulation can be achieved. On the one hand this is challenging because of the large set of heterogeneous data interfaces, which even increase for testing highly automated systems. On the other hand, this approach has several advantages. The main advantages are that functions can be developed in tools that are optimized for a specific purpose (e.g. automated driving, automated harbouring for ships, automated taxing for airplanes, system dynamics, thermodynamics) and system complexity can be reduced.

In the following sections we describe major project results and further challenges.

2.4.1 Simulate automated system functions in various scenarios and environmental conditions

For simplicity of reading, this section focuses on the application of automated driving. Nevertheless, the results were successfully applied in different application domains. Nevertheless, the results were successfully applied in different application domains.

Automated cyber-physical systems functions require environment simulations to create virtual sensor input. Traditional development does not rely on environment sensors. So, this is a completely new requirement. The main task of the environment simulation is the generation of realistic ground truth data based on the selected scenario. The output of the environment simulation varies from general simulation data, such as ego vehicle speed, to simple and complex object lists and beyond to realistic raw sensor data. Automated driving functions require environment simulations to create virtual sensor input. Traditional vehicle development does

not rely on environment sensors. So, this is a completely new requirement. The main task of the environment simulation is the generation of realistic ground truth data based on the selected scenario. The output of the environment simulation varies from general simulation data, such as ego vehicle speed, to simple and complex object lists and beyond to realistic raw sensor data.

In order to support the third layer of the Generic Test Architecture (*Test Execution Platform*), the following requirements had to be fulfilled:

- Work with large data sets (environment simulators provide large data sets)
- Simulate with complex data types like raw data images or object lists
- Extend tool interfaces for coupling environment simulation applications and function simulations for a common simulation
- Specification of the content of environment and sensor data as well as a definition of its meaning

Based on these requirements we extended existing (co)-simulation and environment simulation tools and methods to support scenario-based co-simulation. To handle the challenge of the heterogeneous interfaces we base our concept on standardized interfaces. Two interfaces are therefore to mention: The Functional Mock-up Interface (FMI)[11] and the Open Simulation Interface (OSI)[12]. Both specifications support our concept of a modular simulation framework in which simulation components can be easily exchanged and integrated.

FMI is a tool-independent, generic interface specification to support the exchange of models between tools and the integration of them into a co-simulation. A simulation unit that implements FMI is a Functional Mock-up Unit (FMU). An FMU consists of a description file and code that can be executed. The description file (modelDescription.xml) specifies the simulation unit and its interface. This includes accessible variables, the model structure (inputs, outputs), as well as further information needed for the coordination of the simulation. To use the simulation data, the FMI specification defines methods for accessing input/output data and for controlling the simulation model. This flexible concept enables an easy integration of simulation models and independence of specific tools.

OSI specifies an interface to describe the data structure (message-based) of virtual perception sensors. It contains an object-based environment description in which ground truth and sensor data are specified. Ground truth data is generated from environment simulation applications and represents the object list in a global coordinate system whereas sensor data represents the realistic sensor data coming from sensors and are input for the automated driving function (cp. Figure 2-12). Sensor data is generated in a sensor model that tries to reproduce the realistic sensor behaviour (see also chapter 2.5).

[11] *https://fmi-standard.org*
[12] *Hanke T. et. Al., "Open Simulation Interface: A generic interface for the environment perception of automated driving functions in virtual scenarios", 2017, http://www.hot.ei.tum.de/forschung/automotive-veroeffentlichungen/, accessed: 2019-02-28, https://github.com/OpenSimulationInterface*

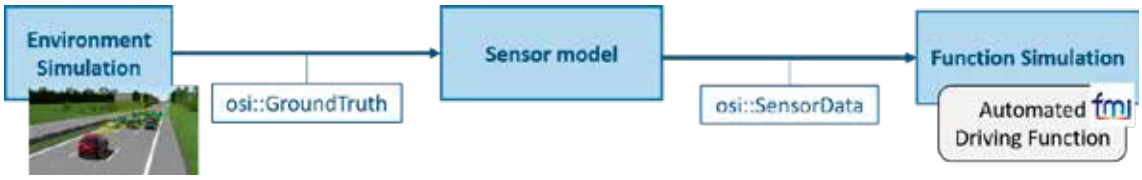


Figure 2 12: OSI interfaces for connecting environment simulation and function simulation

With OSI the compatibility between automated driving functions and a variety of environment simulation applications is made possible. In ENABLE-S3, we implemented OSI-support in several tools (e.g. VIRES VTD, AVL Model.CONNECT) and applied it to several use cases. Based on our experiences, we provided feedback to the OSI community.

Further reading

- ENABLE-S3 deliverable "D58 (D3.22) D3.4.3 v2 ACPS integration platform for online/offline model coupling", to be published on the official project website https://www.ENABLE-S3.eu/media/deliverables/
- Blochwitz et.Al., "Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models", In: Proceedings of the 9th International MODELICA Conference, 2012, pp. 173-184
- Van Driesten C. and Schaller T., „Overall Approach to Standardize AD Sensor Interfaces: Simulation and Real Vehicle", In: Bertram T. (eds) Fahrerassistenzsysteme 2018, Proceedings, Springer, 2019, pp. 47-55

2.4.2 Simulate automated driving functions with real hardware

The development and test of functions typically starts with simulation models which are later implemented in software running on real hardware. Hence, during the development, the simulation models are step-by-step substituted with real software (SiL) and real physical components (HiL) until finally the whole system is tested. This requires a safe simulation environment that can communicate with hardware in real-time and provides safety mechanisms in order to avoid damage in hardware. Therefore, the integration platform must support (real-time) communication technologies (e.g. CAN, EtherCAT) to exchange data between simulation models and hardware platforms.

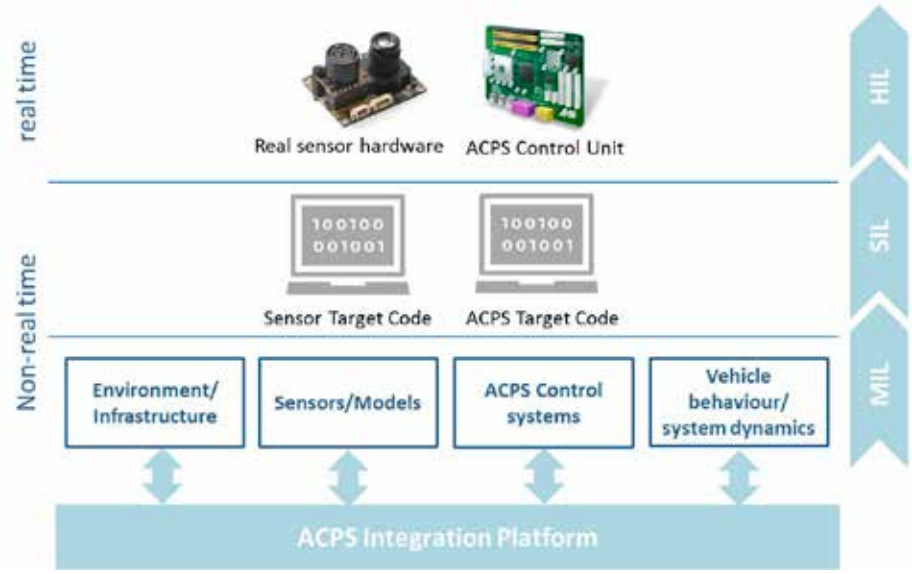


Figure 2 13: Integration platform supporting MiL, SiL, HiL

ENABLE-S3 worked on improved integration methods so that a safe communication between the simulation framework and different physical components is guaranteed. This includes the application of advanced algorithms that predict future values if data cannot be provided in time, control messages that bring the hardware into a safe state in case of errors and concepts for improving the performance. For testing highly automated cyber-physical systems, methods are investigated that integrate real sensors into the (co-)simulation.

We tested automated functions in several use cases from different domains using not only simulation components but a combination of virtual and real physical components. With the modular approach of the integration platform we can easily exchange simulation components with physical components and have hence less integration effort in a later development stage. As a result, we have several working demonstrators where hardware like vehicles, control units or x-ray machines are integrated into the co-simulation.

2.4.3 Distributed Simulation – Simulation Across Companies

Distributed simulation is important when collaborating across companies (e.g. simulation with OEM and supplier) or (simulation) components are located at different locations. This requires a lot more technical integration effort as the communication layer must be defined and setup, communication delays and errors need be handled and the intellectual property (IP) has to be protected. Further, simulation components have to be configured properly in advance and the

coordination of the simulation has to be agreed (who is master and who is slave).

Within the project we adapted existing tools and methods so that environment simulation data can be exchanged also in a distributed network. We improved the workflow for setting up a distributed co-simulation and adapted tools and their interfaces to overcome technical limitations. Based on real demonstrators we analysed timing behaviour (e.g. delays, death times) to improve the master algorithm, which is responsible for the coordination of the co-simulation, in the integration platform. Now, we have an improved algorithm in the integration platform that better manages timing challenges. We have a demonstrator that uses co-simulation to integrate environment simulation with function simulation via a large distance in non-flat networks.

Further, we gained experiences with the Distributed Co-Simulation Protocol (DCP) which is an interface specification that supports distributed co-simulation and the integration of hardware. This specification helps to make the integration platform more modular and flexible also for distributed co-simulation setups. The first public available version has been released in March 2019 but we had access to pre-release versions of this specification. We analysed DCP and implemented it in two tools to exchange sensor data like object lists. First experiences show that with the implementation of DCP some challenges in distributed co-simulation (e.g. configuration of co-simulation) can be handled better. Further, this specification supports real-time simulation and hence, the integration of hardware. More information concerning DCP can be found at https://dcp-standard.org/.

2.4.4 Validation of the simulation environment

Another aspect which is important but has not been covered in detail within ENABLE-S3 is the validation of the simulation environment. This means that we also need to make sure that the simulation provides meaningful results. We consider this as an important aspect for future work. Nevertheless, some fist experiences have been made within ENABLE-S3 as described in Chapter 2.4.7 (Application to Use Case “Automated left Turn at Crossing”).

2.4.5 Align methods between domains

Though co-simulation methods and tools are applied in several domains, the used tools and approaches can differ within domains and even more across domains. However, common requirements exist and offer the possibility to learn from each other. In order to enable an active exchange of information and discuss possible applications of existing co-simulation methods and tools, we organized workshops in which we aligned concepts and approaches.

Of course, there are several differences between domains. Nevertheless, we discovered common requirements such as timing requirements for the integration of hardware or configuration of co-simulations. For the elaborated challenges and requirements, partners of the different domains shared their approaches and thus fostered cross-domain knowledge exchange. Co-simulation standards like FMI or DCP have been developed within certain domains, but can be used across domains as they are not domain-specific. Experiences and application guidelines for those specifications can be shared and enable an easier adoption in other domains. Further, we started to extend tools so that they are usable in more than in one domain (e.g. Model.CONNECT, PhyWise). This is an opportunity for tool vendors to broaden the market.

Methods and best practices for co-simulation exist in all application domains. Within the project we had a workshop to discuss challenges and best practices across domains. Partners from Maritime, Automotive and Health domain participated to exchange information and discuss how to advance methods and tools. Here, we describe the results and discussion points of the alignment between Automotive and Health exemplary.

Differences between application domains using Automotive and Health as an example

Before starting the discussions about methods and tools, we had to establish a common understanding of the simulation goals and concepts based on the use cases. We discovered the following main differences:

- Terms are used differently. To understand the different approaches, we had to find a common terminology. For example, in automotive model management refers to a model database which can be used for the configuration of a co-simulation. In the health domain, model management means the connection of models to get a whole system simulation. Hence, before we could align the methods, we had to align the wording.

- Testing different system configurations versus testing different test cases on the same configuration. In automotive usually several test cases are used to test a system configuration in contrast to Health. In Health a set of test cases exist, which should be executed on different system configurations.

- Event-driven simulation versus time-driven simulation. In Health, the co-simulation is event driven whereas in Automotive time driven simulation is dominant. This requires different coupling strategies and hence, an adaptation of the co-simulation platform.

Common challenges in different application domains

Nevertheless, we discovered several similarities, related challenges and requirements that we can work on together.

- Integration of hardware into the co-simulation. The integration of hardware is required in various application domains. Therefore, coupling strategies and interfaces can be aligned.
- Hard timing/performance requirements. Several co-simulation scenarios require to guarantee hard real-timing requirements to ensure safe results.
- Save state of a simulation. Several applications would benefit from the possibility to save simulation states to perform a rollback.
- Improved simulation model management. Improved model management including version, variability, and configuration management is needed in many applications.

2.4.6 Sustain project results in standards

The results of the elaborated co-simulation approaches to enable the scenario-based verification and validation of highly automated cyber-physical systems makes it necessary to extend and adapt existing specifications as additional requirements have been defined. Therefore, we applied existing specifications and elaborated requirements that have to be fulfilled by standards like FMI, DCP, and OSI, in future. FMI is a well-established specification (cp. Section 2.4.2). Many tools support FMI and it is widely accepted in industry. However, this specification must be extended to support the exchange of complex data types such as object lists or images what is needed to connect environment simulation applications to the co-simulation. Within the project we developed several demonstrators in which we exchanged sensor data. Based on these experiences we collected requirements the FMI specification has to support. Partners of the consortium are involved in the FMI development and brought in the requirements from our project and provided a tentative solution.

The ACOSAR project, which developed a generic specification for distributed co-simulation, partly overlapped in time with the ENABLE-S3 project. The result of this project was the Distributed Co-Simulation Protocol (DCP) specification that in the meantime has been published via the Modelica Association[13] (like FMI). Requirements from the ENABLE-S3 project have been considered for the specification during project time. In return, we evaluated the DCP specification in ENABLE-S3. As mentioned above, we implemented the DCP to exchange simulated sensor data between two platforms, which represents one of the first DCP implementations for complex data types.

[13] *https://www.modelica.org/*

For connecting environment simulators and function simulators, the Open Simulation Interface (OSI) is important as it defines first the data (e.g., object lists with position, size, etc.) that needs to be exchanged. Secondly, it defines also the semantic meaning. For example, OSI defines in the environmental conditions that fog dense means that a sensor has a maximal range of 50 meters. The definition of the contents is challenging as this should be a generic specification that should be supported by various environment simulation applications. Within the project we applied OSI for several demonstrators (cp. Section 2.4.7) in various use cases. Hence, we gained some experience using OSI. We summarized these experiences in form of requirements that we described on the official OSI Github page[14]. Through a close cooperation with the PEGASUS project[15], which enhances the OSI specification, we further could directly discuss our experiences with partners from this project. Consequently, our project results are integrated into the OSI specification.

The cooperation with standardization groups and projects was a major contribution to sustain our project results. ENABLE-S3 brought in requirements and proposals so that the verification and validation of automated driving functions is supported by specifications. With generic specifications and standards, it is possible to facilitate integration of simulation components and hence, provide an improved interoperability. At the end this helps to save costs.

2.4.7 Application Examples

Connect environment and function simulator using OSI

The integration of environment and function simulations has been applied in many use cases. A major goal of the integration platform is the use of generic, standardized interfaces to allow a flexible and modular integration of simulation components. OSI has been applied in several use cases, namely the highway pilot, intersection crossing and valet parking. Here, the highway pilot is demonstrated.

In order to evaluate and apply OSI, the respective tools had to be extended by an interface implementation. For the highway pilot demonstrator, we implemented an OSI interface for VIRES VTD, AVL Model.CONNECT and further at the Robot Operating System (ROS) in which the OSI data is visualized.

The demonstrator is distributed across three machines. On the first machine, VIRES VTD generates the ground truth (cp. Figure 2 14). On the second one, the co-simulation platform Model.CONNECT is running, that acts as simulation master and coordinates the whole simulation. Moreover, a simple sensor model has been implemented that should represent realistic sen-

[14] *https://github.com/OpenSimulationInterface*
[15] *https://www.pegasusprojekt.de/en/home*

sor behaviour instead of a perfect sensor. This sensor model as well as the automated driving function, an ACC-function model, and the vehicle dynamics are executed on the same machine as Model.CONNECT. On the third machine, the ROS visualization component is running. An overview of the highway pilot test setup can be found in Figure 2 18. This illustration shows all simulation components that are connected via generic interfaces.

Figure 2 14: Environment simulation with VTD, providing 'ground truth' sensor information

TCP/IP communication is used to connect machine 1 and 2. For the synchronization and exchange of vehicle simulation data, the VTD proprietary RDB format is used. The ground truth data is transmitted in OSI format. In Model.CONNECT™ a binary port has been implemented that supports OSI sensor model packaging[16]. This enables the integration of the sensor model as FMU. The sensor model converts the OSI::GroundTruth, received form the environment simulation, to a OSI::DetectedObject list (cp. Figure 2 15). The behaviour of the model includes a transformation to relative coordinates, with respect to the ego car and a reduction of the range based on environmental conditions like precipitation, fog and sensor attributes like the field of view. The output of the sensor model is connected to the ACC function, implemented as FMU, which is furthermore connected to the vehicle dynamics to override throttle and brake pedal, in case the ACC function is activated. Finally, OSI::GroundTruth and OSI::SensorData are transferred via a TCP interface to the ROS framework which represents the interface between machine 2 and 3. In ROS the ground truth data and the sensor data are visualized and evaluated.

[14] *https://github.com/OpenSimulationInterface/osi-sensor-model-packaging*

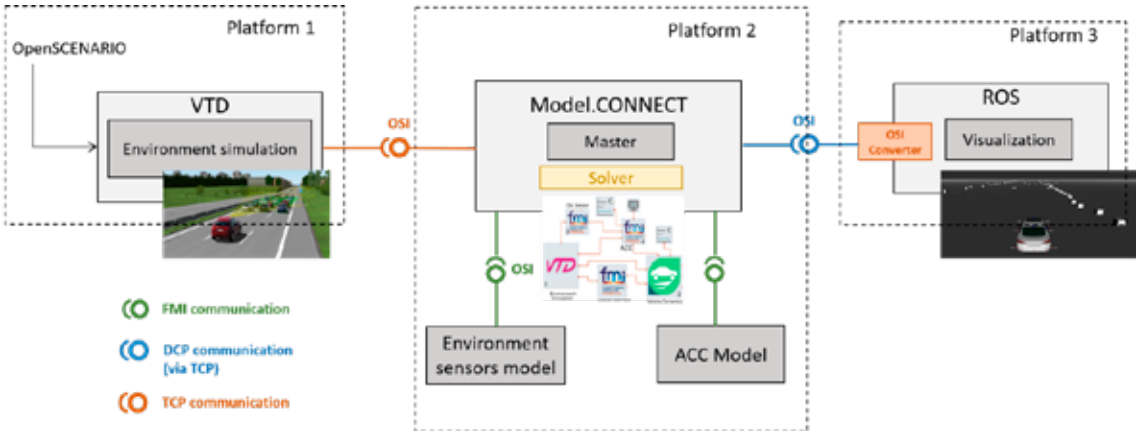


Figure 2 15: OSI interfaces in reference simulation architecture

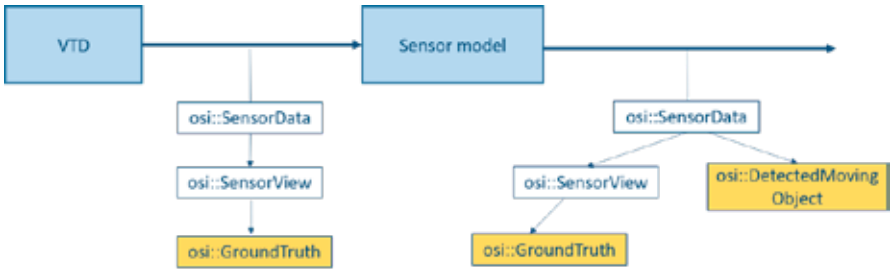


Figure 2 16: OSI messages

The use of generic, standardized interfaces improves the interoperability of the integration platform. This enables for instance, that the sensor model can be integrated into Model.CONNECT but also into VIRES VTD without changing the interface. Further, sensor models can easily be exchanged or implemented in other tools supporting OSI. For example, a second sensor model has been implemented in Cassandra which can easily be integrated into this demonstrator. This increases flexibility and modularity and in return it reduces integration effort and hence, costs.

More information concerning OSI and our experiences with this interface as well as the demonstrator can be found in the publication[17] and public deliverable of the ENABLE-S3 project[18].

Simulate Highway pilot functionality with real hardware

The integration of hardware into the co-simulation requires additional effort. In the project several demonstrators exist that connect hardware to the simulation. For the use case highway pilot, a chassis dyno, which is used to fixate and operate a passenger car, has been integrated into the simulation. The car includes autonomous driving functionality and sensors for environmental perception. The used simulation components, real and virtual, are summarized based on the generic test architecture in Figure 2 17.

[17] *Marko N., Rübsam J., Biehn A. and Schneider H., "Scenario-based testing of ADAS: Integration of the Open Simulation Interface into co-simulation for function validation", submitted*
[18] *58 (D2.33) - D3.4.3 v2 ACPS integration platform for online/offline model coupling, see ENABLE-S3 website: https://www.ENABLE-S3.eu/media/deliverables/*

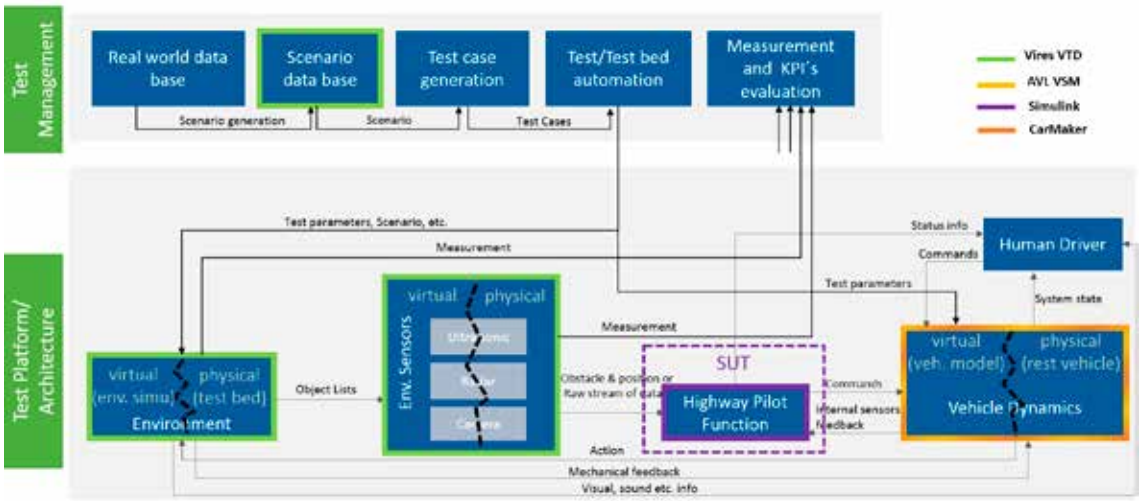


Figure 2 17: Highway Pilot test architecture

The simulation setup consists of virtual and physical components that can be applied together. The integrated physical components are:

- **Environment model:** The physical environment model is a chassis dynamometer. It is used to reproduce plausible, physically correct environmental conditions for the vehicle and therefore for the autonomous driving function. The test bed includes dynos, vehicle fixation and hardware interfaces for the sensor stimulators (see Figure 2 18).

- **Perception sensors:** The physical sensors (five radar sensors and one mono camera) are integrated in the vehicle. Not all sensors have been stimulated in the test setup.

- **Sensor stimulators:** The environmental sensor stimulators are used to connect the virtual and physical environment (respective the physical vehicle). They convert the information from the virtual environment, such as object lists or video images, to hardware readable data, which is converted to physical information (radar waves and light waves).

- **Vehicle model:** To integrate vehicle behaviour into the simulation, a physical vehicle is used on the test bed. The vehicle is equipped with multiple sensors and cameras. It is mounted directly on the wheel hubs on the test bed. Torques onto the wheels can be applied by the dynos of the test bed. The optical and radar sensor information is coming from the sensor stimulators.

These physical components together with some virtual components are co-simulated, so that the Highway Pilot function receives raw sensor information from the vehicle sensors such as distance to obstacles and raw video data. This information is interpreted by the Highway Pilot function that generates demand actions such as acceleration demand and steering demand to the vehicles engine and steering system.

Figure 2 18. Chassis dynamometer setup of the Highway Pilot

During the project, the DrivingCube has been extended by multiple components. A lot of effort has been put into the improvement of the setup, bug fixing and performance enhancements. The co-simulation runs stable and reproducible and the expected results in performance and usability could be achieved. Various tests were performed in Graz, Bensheim and Karlsruhe. However, the sensor stimulation components for radar and camera stimulation are still under development[19],[20].

Application to Use Case "Automated left Turn at Crossing"

This example shows a first evaluation of shifting V&V from physical to virtual testing by executing comparable test cases on proving ground as well as in a Vehicle-in-the-Loop (ViL) environment. The overall goal was to assess how good the simulation reflects reality. The same system was used in both environments. We evaluated the approach by comparing Key Performance Indicators (KPIs), calculated based on collected data. We give a brief overview of the method that we used:

1. We selected scenarios related to left turning at two-lane city intersection to validate the highly automated cyber-physical system.

2. We planned test driving in two environments: proving ground and simulation lab. In both studies, we designed traffic conditions at the intersection to challenge the driving task of the autonomous car. Additionally, we acquired driving data during test driving.

3. We calculated and analysed KPIs with data acquired from the test driving.

[19] *D4.1.2 v3 Demonstration Report + Feedback from the automotive use cases*
[20] *D3.4.3 v2 ACPS integration platform for online/offline model coupling*

Relevant Scenarios

A list of typical scenarios for left turning at intersection was defined as illustrated in Figure 2 19. They include (a) another vehicle crossing before turning left, (b) another vehicle blocking the line of sight to the base station while a vehicle is approaching the intersection from the opposite direction, and (c) two vehicles approaching the intersection where the first one is blocking the line of sight to the second car. In each scenario there is at least one incoming vehicle approaching from the opposite site of the intersection. Since both vehicles (the vehicle of interest and another vehicle) meet at the centre of the intersection, the driver of the test vehicle has to decide to turn before or after the incoming vehicle. Furthermore, the distance between the vehicles has different levels, such as small gap, medium gap, and large gap. Thus, each of the scenarios bears different challenges for the autonomous system.

In summary, there were ten different traffic conditions: (SC0) Baseline (turning without traffic), (SC1) Baseline-autonomous, (SC2) Crossing Vehicle – small gap, (SC3) Crossing Vehicle – medium gap, (SC4) Crossing Vehicle – large gap, (SC5) Laser Blocked – none traffic, (SC6) Laser Blocked – small gap, (SC7) Laser Blocked – medium gap, (SC8) Laser Blocked – large gap, (SC9) Two vehicles incoming – small gap and (SC10) Two vehicles incoming – large gap.

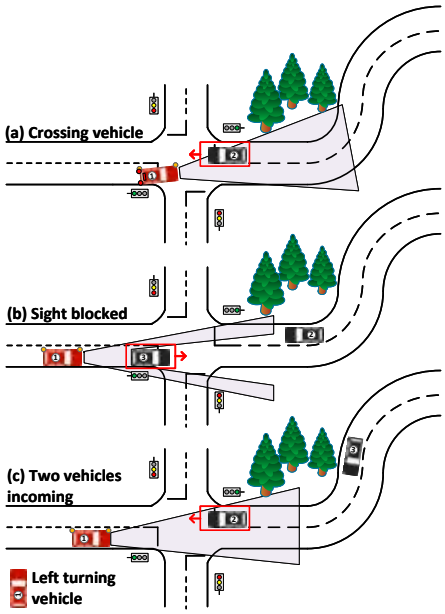


Figure 2 19: Traffic conditions in left turning scenario.

Test Platforms and Test Driving

The proving ground consists of a single intersection at the traffic training area[21] located in the North of Braunschweig. The intersection has four legs with the main street running east-west and the branch street running north-south. The approaching lanes on the four legs are split into two lanes to accommodate for left turning.

The VR-Lab is a highly dynamic and scalable simulation environment providing a 360° field-of-view with high resolution visualisation. The flexible integration of different mock-ups and vehicles allows the testing and evaluation of automation and assistance functions through repeatable and reproducible scenarios. By using a common software framework, it is possible to link it with other simulators and infrastructures.

[21] *Verkehrsübungsplatz Braunschweig e.V. http://www.vp-bs.de/*

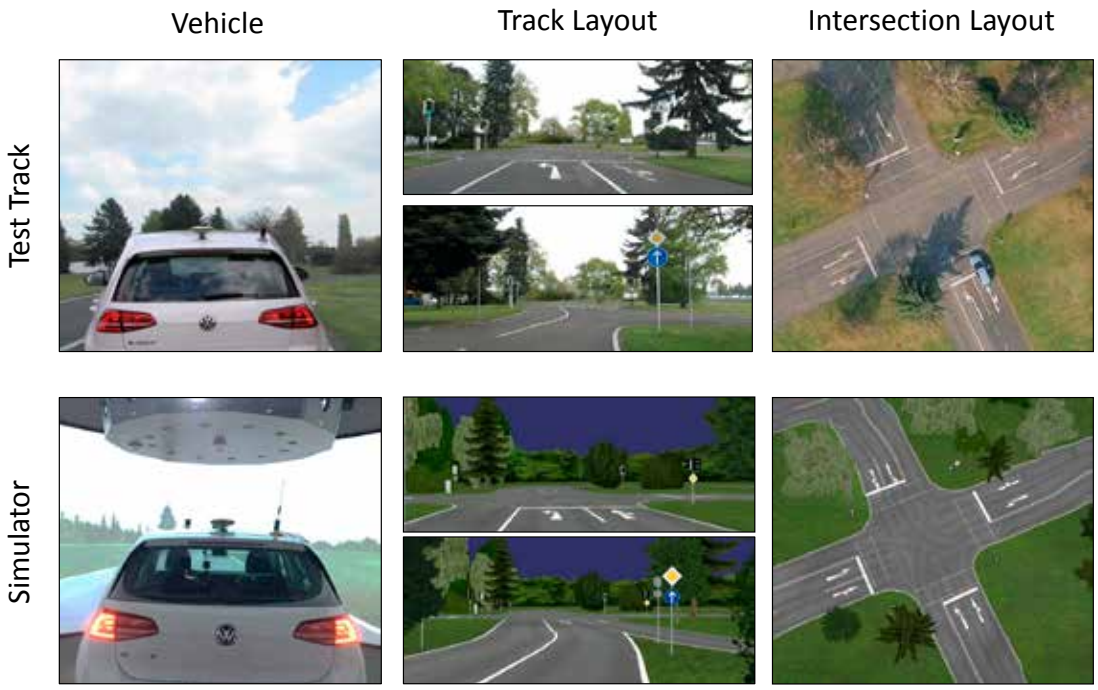


Figure 2-20: Comparison of real world (upper row) and simulator visualization (lower row)

Key Performance Indicators (KPIs)

We calculated all KPIs (see Table 3 and Figure 2 21/22) on the basis of the data acquired from the test driving on proving ground and on simulation.

Table 3: List of KPIs

KPI	EXPLANATION
Stops	number of times the car stopped in intersection
Time Stops	cumulative time of critical stops
Maximum Lateral Acceleration	Maximum lateral acceleration
Maximum Longitudinal Acceleration	Maximum longitudinal acceleration
Maximum Lateral Jerk	Maximum lateral jerk
Maximum Longitudinal Jerk	Maximum longitudinal jerk
Travel Time	time spent on intersection
V At Entry	velocity when entering the intersection
V At Exit	velocity when exiting the intersection
g_at_exit	distance to lane center at exiting intersection
d_min_left(right)_entry	Minimal distance to lane marking at entry arm
d_min_left(right)_exit	Minimal distance to lane marking at exit arm
d_min_virtual_lane	Minimal distance to virtual lane marking within intersection
RMS normal distance to ideal track	Root mean squared distance between position and ideal trajectory
RMS_of_longitudinal_jerk	Root mean squared of longitudinal jerk
RMS_of_lateral_jerk	Root mean squared of lateral jerk

A. KPIs to judge safety ("Observables Type a"):

time_Start to End
d_min_left(right)_entry
d_min_left(right)_exit
d_min_somewhere +
s_on_ideal_track_position
a_at_exit
v_at_exit
RMS normal distance to **ideal track**

B. KPIs to judge comfort ("Observables Type a"):

Max jerks in vehicle coordinate system (al 3)
RMS values of all 3 jerk values

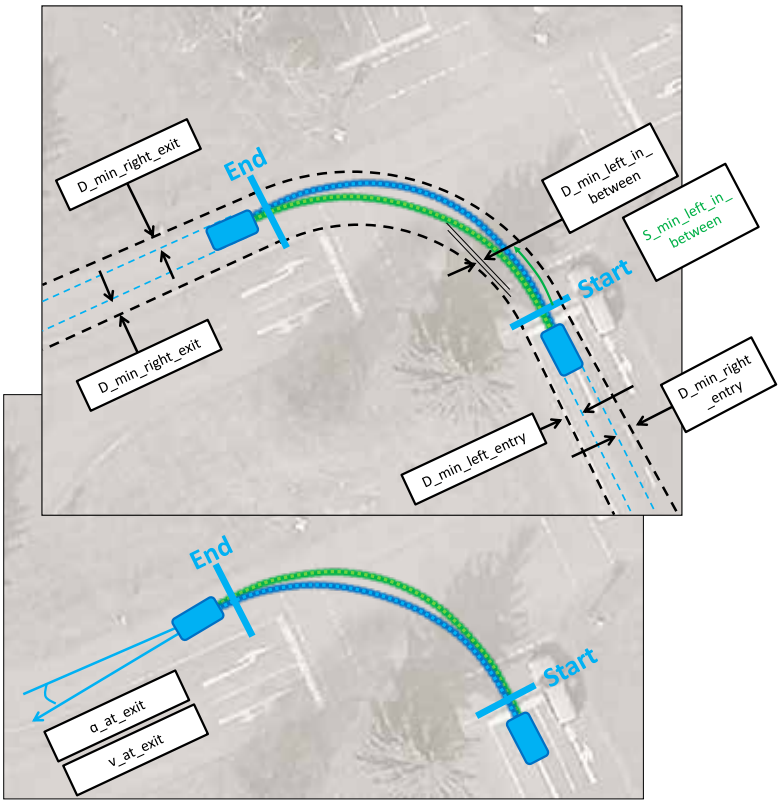


Figure 2 21. KPIs for left turn scenario

We use the KPI "velocity at exit" to illustrate the approach. In Figure 2 22, the velocity is plotted relative to the standard trajectory. Dashed lines represent the entry and exit points of the intersection. Figure 2 21 shows that the velocity at exit was evenly distributed with a tendency to being higher on the proving ground.

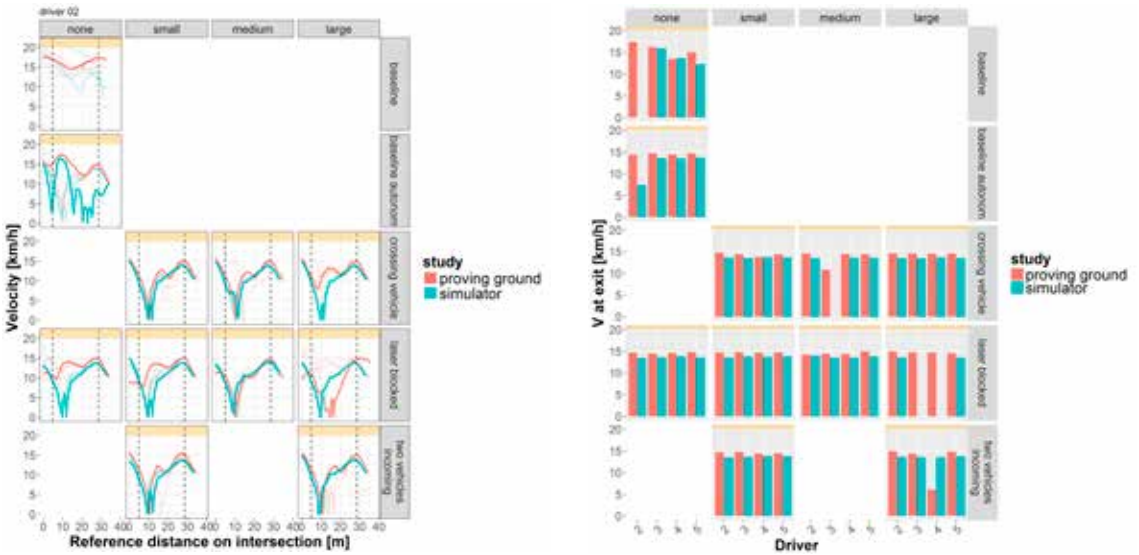


Figure 2 22: Visualization of the KPI related to velocity. Left: velocity profile of driver. Right: velocity at exit, all conditions.

Further reading

- Marko N., Rübsam J., Biehn A. and Schneider H., "Scenario-based testing of ADAS: Integration of the Open Simulation Interface into co-simulation for function validation", submitted

2.5 System Component Simulation and Stimuli

Key Results:

- By functional decomposition of perception sensor systems into their components, we have commonly defined generic consecutive functional blocks as subsystems of these sensor systems. This allows to implement generic physical models of the different functional blocks, as already done within ENABLE-S3 for the different sensors, to gain better analysis of possible misbehavior of test automated systems.

- We have suggested generic interface definitions for the different sensor models (radar, lidar, camera) that connect the functional blocks. They improve modularity and thus reusability of the implemented sensor models. Virtual verification and validation of automated systems depend on proper perception sensor models adapted to different testing objectives. Now, the integration of sensor models in different test environments can be done much more efficiently and specifically.

- We have developed perception sensor simulation for different types of sensor systems (e.g. radar and lidar) and made advancements in the field of sensor modeling. Before ENABLE-S3, virtual validation was limited, because of the absence of appropriate sensor models. Now, it is possible to reflect different sensor characteristics on different data processing levels in simulation.

- We have developed approaches for perception sensor stimulation (e.g. mixed reality lidar, radar stimulator) for different types of sensors. Before ENABLE-S3, a mixed virtual and real validation of automated systems was limited by the ability to physically stimulate perception sensors from a virtual environment, which is now possible.

- We have developed solutions for communication channel simulation (e.g. wireless communication simulation, V2X channel emulator). This enables the detailed analysis of system behavior under real-world conditions and events.

As mentioned before, sensor models are an integral part of the Test Execution Platform and are one of the main new requirements for virtually validating automated cyber-physical systems. The development and validation of sufficiently detailed sensor models is still a main challenge and has therefore been handled in a separate work package. This section summarizes the results of this work package, which has mainly been focused on defining the functional decomposition of the most relevant perception sensor systems. This section is therefore a more detailed description of the concepts described in Section 2.4.1.

2.5.1 Interface Definitions for Sensor Models

Perception sensor signal and data processing consists of several consecutive steps, starting from analogue waveforms propagating through a medium towards a machine-interpretable representation of information that is finally obtained from the perception sensor system. From a modelling perspective, a modular structure of the models is beneficial to meet different simulation objectives. This is motivated by the following requirements:

1. Different applications of highly automated systems require different data to be reported by a perception sensor system. When understanding a sensor (model) as an encoder of information from a (virtual) scene, a modular approach comprising several different processing steps gives more degree of freedom when stimulating the function or system under test.

2. A sensor model must capture all potential sensing errors, which the safety architecture has considered as harmful for the overall safety. A modular sensor model allows investigating, whether the model captures sensor errors correctly. For this purpose, consistency checks might be performed throughout the pipeline of data processing, which is possible with a modular approach.

Currently, these different requirements and abstraction levels affect the reusability of sensor models that need to be tailored for each specific application. To overcome this limitation and to create a modular approach, generic interfaces (IF_n) across common sensor types have been defined in dedicated working groups. This allows a fast understanding of implemented abstraction levels as well as integration in environment simulation platforms. A typical object detection and classification process is chosen to serve as the basis for decomposition. The defined interfaces are listed in Table 4. As the chosen interfaces for simulation of the sensors are different, because of differently determined generic functional blocks with respect to their stage in the signal and data processing, the interfaces have been numbered differently for radar, lidar and camera.

Table 4: Identified interfaces of radar, lidar and camera sensor systems

IF	Radar	IF	Lidar	IF	Camera
1	Transmitting Antenna Analog, time-domain signal, RF-Signal Physical @ GHZ		(Emission)	0	Visual Path Day, Night, Weather, Artificial Light, ...
2	Receiving Antenna Analog, time-domain signal, RF-Signal Physical @ GHZ		(Reception)	1	**Optic** Camera Model, Projection Parameters, Distortion Co-efficients
3	Beat signal ADC output, Digital, time-domain signal @ MHz	0	Raw signal ADC output, digital, time-domain, after thresholding	2	Image and Grabber Sensor Format, Frame-timing, Sensitivity, ...
4	Spectral periodogram Digital spectral signal			3	Image Buffer Header and Image Data
5	Peak list List with a finite number of entries, Signal level, location	1	Raw scan	4	Filter Output Depends on ADAS function
			List of reflections in spherical coordinates, single sensor, incl. intensity, etc.		
6	Target list List with a finite number of entries, Peaks associated to targets, removed ghosts or artefacts from signal processing	2	Point cloud List of reflections in (mostly) Cartesian coor¬dinates, single or multiple sensors, incl. intensity, etc.	5	Feature List Edge, Corner, Circle, Dynamic Data, Colour Processing
7	Object list List of tracked and classified objects with estimated properties like size, orientation, speed, etc., single or multiple sensors	3	Object list List of tracked and classified objects with estimated properties like size, orientation, speed, etc., single or multiple sensors	6	Object list List of tracked and classified objects with estimated properties like size, orientation, speed, etc., single or multiple sensors

Exemplary functional blocks of radar sensor systems

The huge diversity in radar models, resulting from the level of physical complexity as well as target applications, is projected into functional blocks that comprise a variety of possible implementations. This reflects the variety of specific radar sensor models while providing a quick and easy understanding of the abstraction level of the radar simulation interface to the application engineer. Except for Interfaces IF_1 and IF_2 that reflect the analogue transmitted and received signal, the functional blocks and interfaces after the analogue-to-digital conversion are visualized Figure 2 23. The interface definitions allow e.g. to create radar sensor models with data output between different function blocks modelling the functionality of the sensor at different levels[22]. With these blocks and interfaces at hand, the later described individual radar sensor models for different purposes and from different partners can be described and aligned.

Figure 2 23: Block diagram of radar sensor system interfaces

Exemplary functional blocks of lidar sensor systems

The identified function blocks and interfaces of the lidar sensor systems, shown in Figure 2 24, allow describing different real and virtual lidar sensor systems as well as modelling methods for data generation on different output interfaces. Here, as for radar and camera, object detection has been chosen for the final goal of the data processing. In case of other possible data processing methods/steps, like for localization and mapping, different functional blocks would be the result. Nevertheless, the method would be the same, as performed for the common task of object detection.

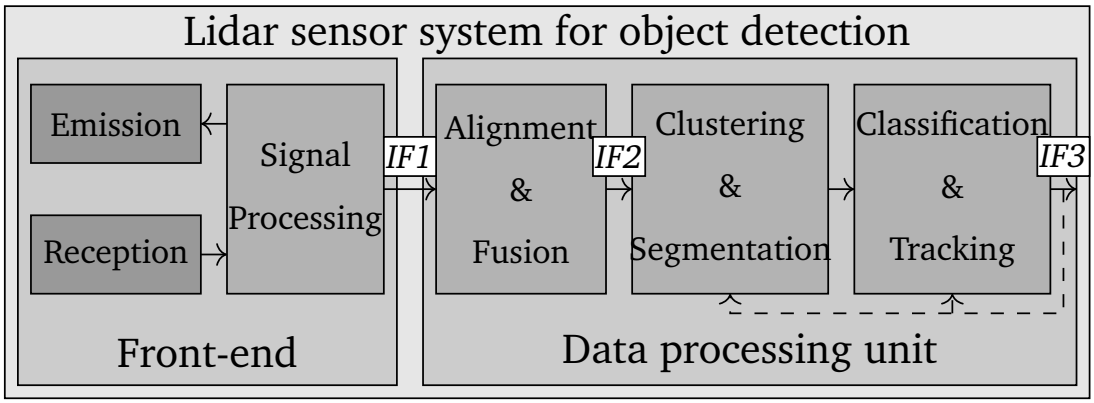


Figure 2 24: Block diagram of lidar sensor system interfaces

[22] *Holder, M. et al.: Measurements revealing Challenges in Radar Sensor Modeling for Virtual Validation of Autonomous Driving, 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, 2018, pp. 2616-2622.*

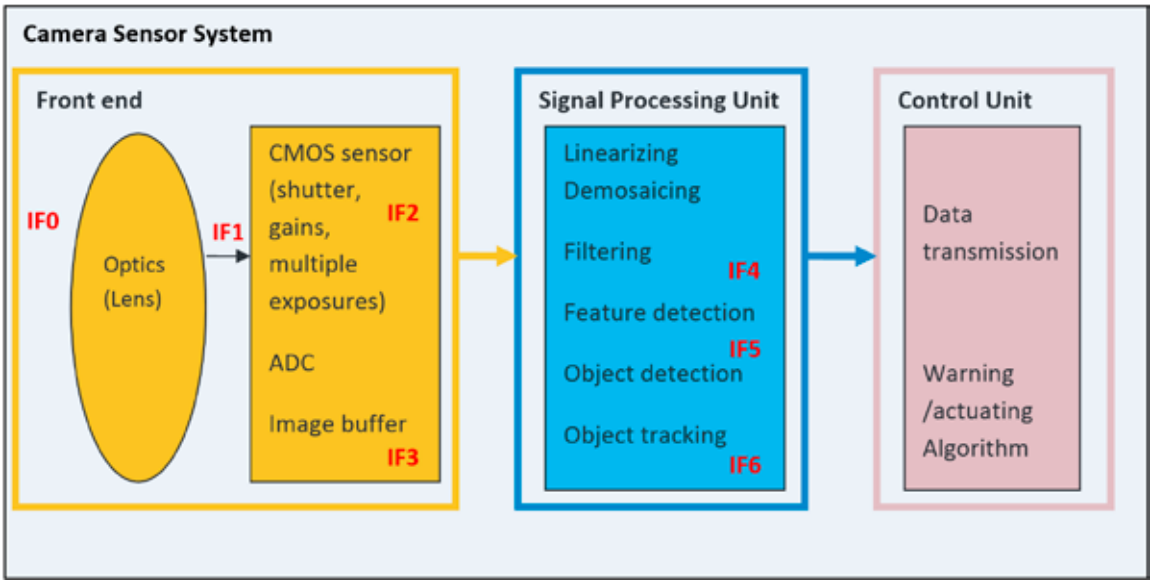


Figure 2 25: Block diagram of a camera sensor system and its interfaces

Validation of sensor system simulation as final objective

A special focus of all partners lies on the validation of the different implemented sensor models. To obtain valid sensor models at the end, with the described interfaces at hand, next possible steps are to find metrics for benchmarking of sensor models on different interfaces and e.g. to compute metrics on data of the same origin interface and further processing for correlation of metrics applied on those subsequent interfaces to investigate error propagation.

2.5.2 Perception Sensor Simulation

As already described in the first subchapter, perception sensor simulation is crucial for testing automated systems. This subchapter shortly presents some sensor models developed within ENABLE-S3 and their approaches.

Multi-output lidar sensor model

A lidar sensor model is integrated in the Valet Parking Test System for SiL/HiL/MiL tests. It is using ray casting to generate realistic point clouds and is based on a collaboration between several partners within ENABLE-S3. It is parametrized to simulate Ibeo LUX 2010 lidar sensors, as used in exemplary parametrization data collection.

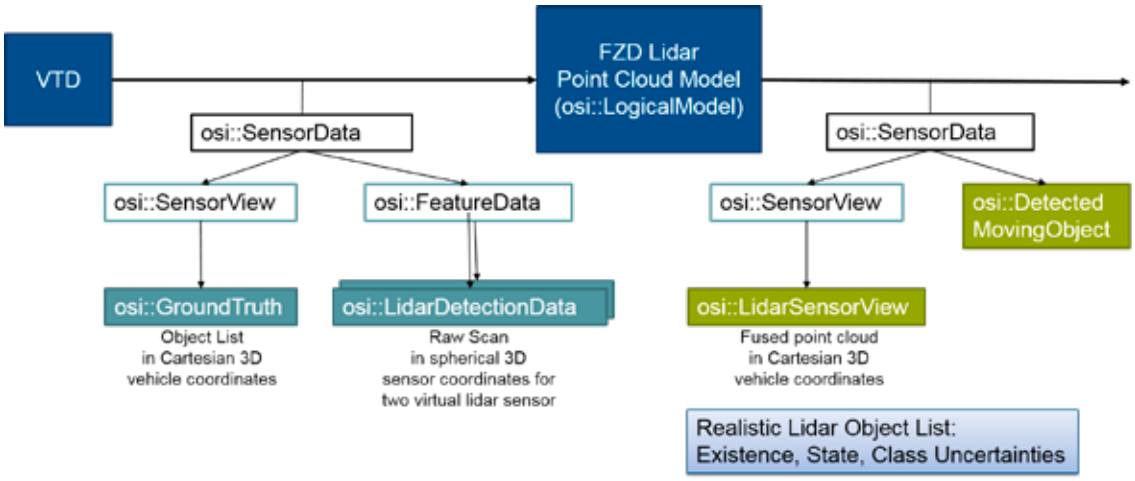


Figure 2 26: Lidar sensor model

The speciality of this lidar model is to output a realistic object list, based on the simulated point clouds, as it is exemplarily visualized in Figure 2 26, while reducing computation effort by using ground truth object information. It covers physical effects on the point cloud data like occlusion, layer orientation, etc. For the object list, existence, state and class uncertainties are modelled. Sample validation is performed by comparing model outputs with real data. The point cloud that is detected by the sensor can also be accessed and visualized.

Therefore, the lidar sensor model is implemented in a modular way so that it can output data on the identified lidar sensor system interfaces IF_2 and IF_3. All interfaces are compliant to OSI3 (see Figure 2 26 and Chapter 2.5.4). Further, error propagation can be investigated, and failures can be injected at different data processing steps. Besides, once the point cloud is obtained, computation effort is low, as we use the available ground truth information instead of algorithms for object classification and tracking. A possible output of the model is visualized in Figure 2 27. This figure shows real sensor output of an Ibeo LUX 2010 lidar sensor system. This real lidar sensor is capable to output data on two interfaces like the simulation model and therefore can be simulated in every aspect, when parametrized accordingly.

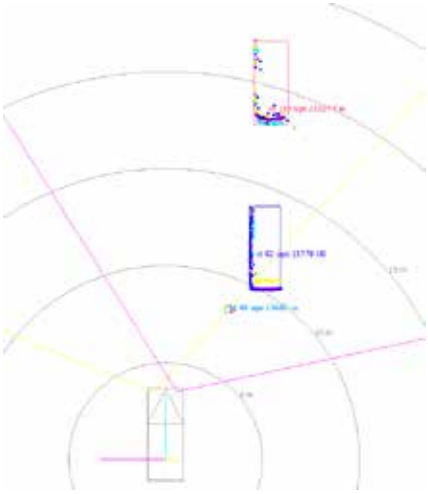


Figure 2 27: Lidar point cloud and object list of Ibeo LUX 2010

Radar modelling approaches

Within the project, several radar sensor models were implemented that demonstrate the variety of applications and implementations including the effective common interface definition. The perception sensor simulation visualized in Figure 2 28 named radar signature and stimulation input generation (RASIG) is used to serve as input for the radar target stimulator described in the following chapter. It is using a standalone ray tracing approach to simulate the radio channel, in terms of propagation and reflection. For standalone operation position and orientation of dynamic objects are transmitted from the environment simulation each frame. At initialization, 3D object geometries and material properties of all surfaces are transmitted.

The result of the channel simulation is converted into a Range-Azimuth-Doppler bins map, with a bin resolution of the simulated (and stimulated sensor). From this range-azimuth-Doppler map stimulation points (r-ϕ-f_d-σ) are computed. The stimulation unit then stimulates the radar sensor over-the-air.

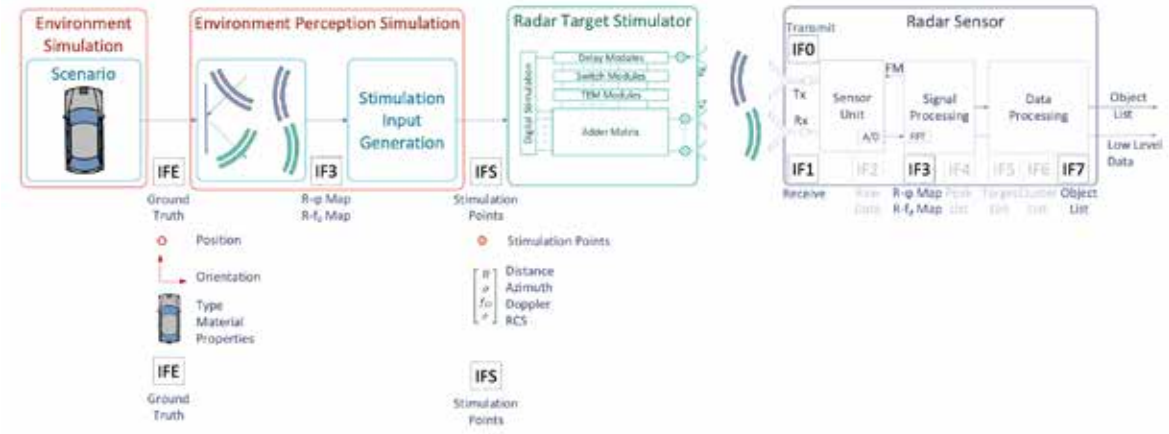


Figure 2 28: Radar sensor model as input for radar stimulation over the air

Further information:

- Maier, F. M.; Makkapati, V. M.; Horn, M. (2018): Adapting Phong into a Simulation for Stimulation of Automotive Radar Sensors, in: IEEE International Conference on Microwaves for Intelligent Mobility
- Gadringer, M. E. et al.: Radar target stimulation for automotive applications, IET Radar, Sonar & Navigation Vol. 12, Issue 10, October 2018, p. 1096-1103
- Maier, F. Michael; Makkapati, Vamsi P.; Horn, Martin (2018): Environment perception simulation for radar stimulation in automated driving function testing, in: e & i Elektrotechnik und Informationstechnik 135 (4-5), p. 309–315

Besides, a phenomenological radar sensor model was developed, as shown on Figure 2 29, based on an automotive radar sensor system. The sensor model comprises physical dimensions and positions of transmit and receive antenna arrays including their elements. This model was integrated into a target list processing radar simulation, whereas the target lists were either inserted manually or generated from object lists that were provided by an external environment simulation. The radar simulation was transferred from automotive to framing domain within ENABLE-S3 to allow first simulations for testing farming machines with radar sensor-based perception[23]. The validation of the radar sensor model was performed at interface IF_3 using pointwise measurements of the physical radar sensor.

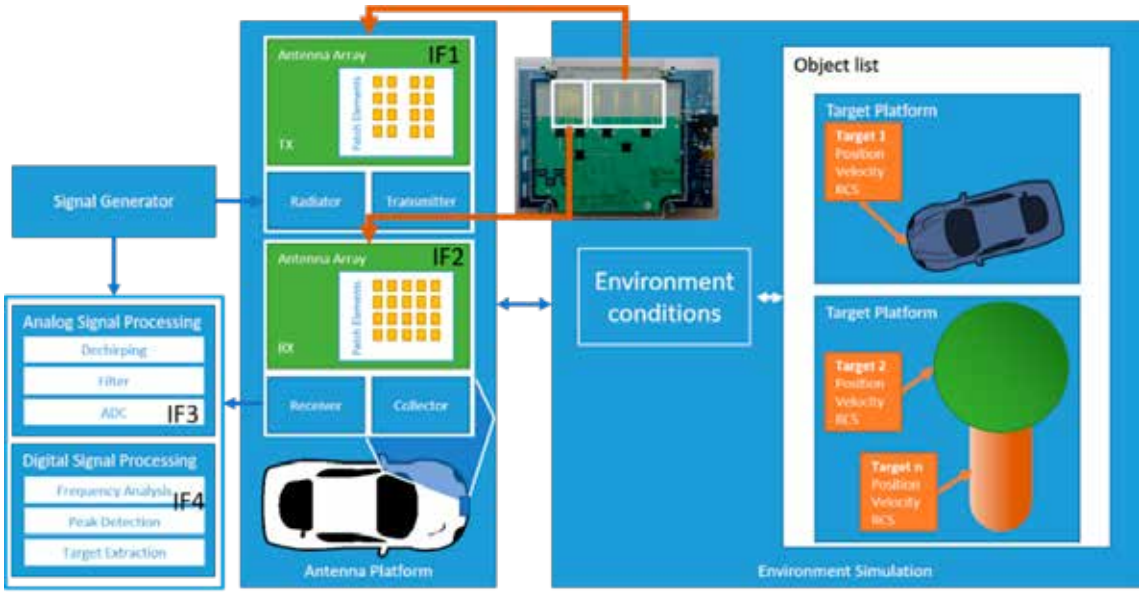


Figure 2 29: Radar sensor model for physically based radar target simulation

Additionally, a real-time capable automotive radar sensor model was created and validated (Figure 2 30). This model focuses on the fast and realistic simulation of target and tracked object lists, which are the interfaces provided by today's state-of-the-art radar systems. Different types of faults are injected to test the reliability of the AV controller, including noise on the measured quantities, quantization artefacts due to the radar cell resolution or CAN-bus transport, clutter detections, target obstruction, multiple echoes per target, tracking phenomena such as track splitting or association, etc. Validation methods of the sensor model were studied based on measured data collected using a real automotive radar sensor in a controlled experimental environment.

[23] *Rooker, M.: Towards improved Validation of Autonomous Systems for Smart Farming. Workshop on Smart Farming, CPS Week 2018.*

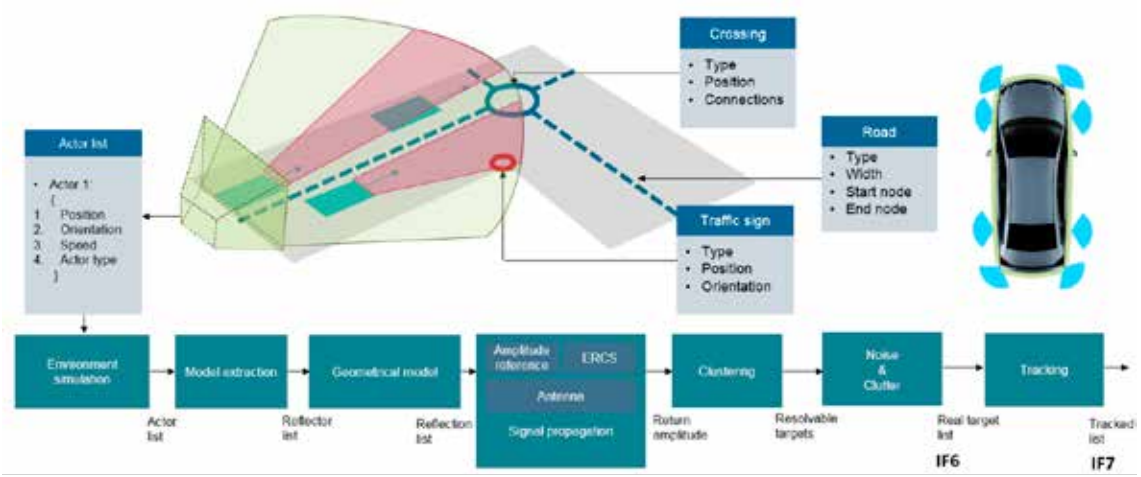


Figure 2 30: Radar sensor model for radar-based object lists

Camera model

The following camera model developed within ENABLE-S3 is integrated in the OSI demonstrator and described in Figures 2-31. It can be applied for failure injection during SiI and HiL testing.

The camera model can be implemented in Cassandra with or without FMU support. The FMU support makes it possible to integrate FMUs from other suppliers. Examples of failure injection are phantom cares appearing as reflections of real cars on a wet road or missing cars, which are not detected due to extreme fog or rain intensity.

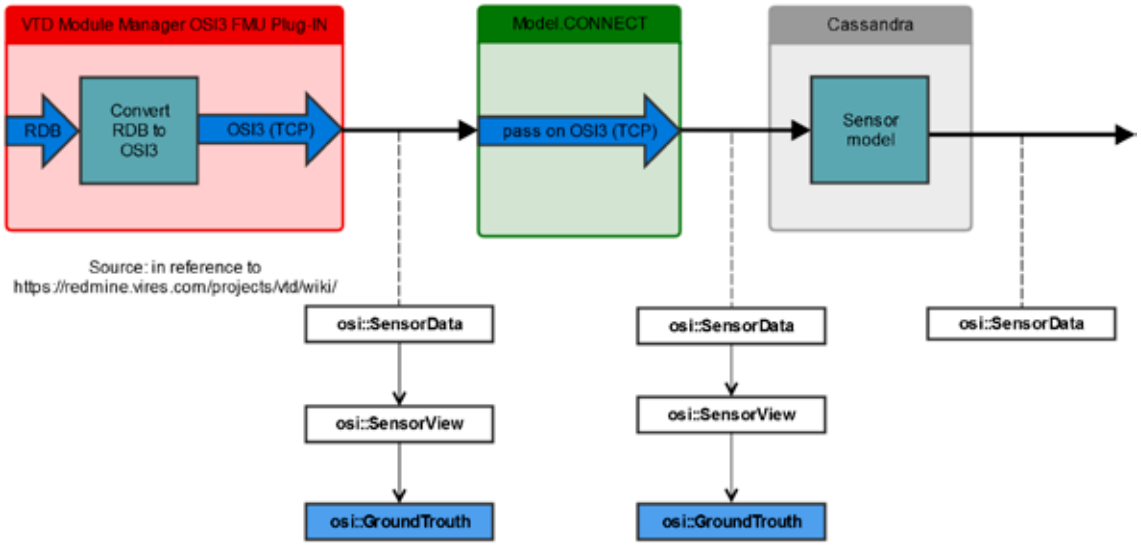


Figure 2 31: Camera model for failure injection

Ultrasonic sensor model

An ultrasonic sensor model was created during the ENABLE-S3 project, as shown in Figure 2 32. The model is used to simulate the processing chain of the ultrasonic sensor at different stages and the corresponding environment representation. First, voltage pulses are sent to an equivalent model of the piezo transducer. This results in sound waves that are transferred to the environment model. The corresponding echoes perceived by the sensor are computed, considering the angular gain of the piezo capsule, the loss due to the distance, environmental noise and wave distortion due to the shape of the target. Multiple echoes and targets are considered at the same time. The received signal is then processed by band-pass filtering, amplifying and thresholding. This results in a set of detections that can be fed in a control algorithm for its validation. Finally, measurements were acquired to parametrize and validate the model of the sensor, including the parameters of the piezo transfer function and the amplitude drop due to distance.

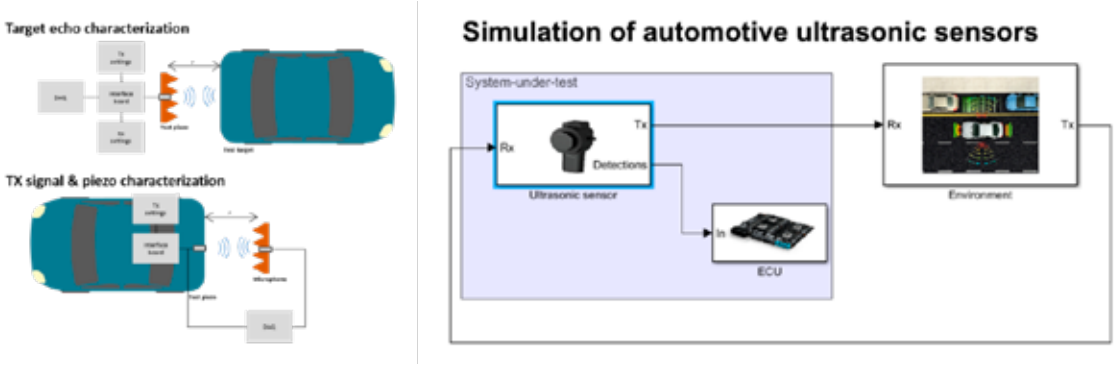


Figure 2 32: Ultrasonic sensor simulation and validation measurements

2.5.3 Perception Sensor Stimulation

Stimulation of perception sensors is needed in ViL tests of automated systems. The system to be tested perceives its surroundings by sensors, implementing the intended functionality. To provide safety, scalability, comparability and reproducibility during tests of such systems, sensor stimulation is a powerful tool.

Sensor stimulation can be implemented on different interfaces of the system and sensors. Stimulations can be performed on hardware level, but also on more abstract interfaces like point clouds, images or even more high-level on object lists.

Mixed reality lidar stimulation

The mixed reality lidar stimulator allows to test lidar based automated driving functions in a safe, reproducible environment. A lidar based automated system can be tested by stimulating its sensor fusion control unit with either virtual lidar scans, or a combination of real scans and simulated lidar data, as shown in Figure 2 33.

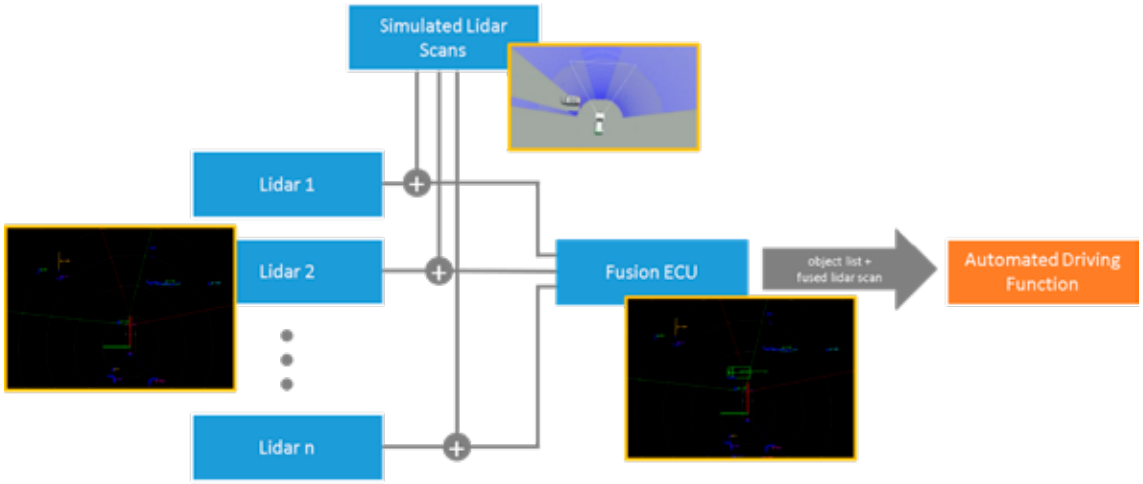


Figure 2 33: Multi lidar sensor simulation and stimulation

This sensor stimulation enables multiple testing, for instance testing and benchmarking of relative localization methods while having ground truth data to compare to. This is achieved by stimulating the localization system with simulated lidar scans, while the real vehicle drives on a free testing ground where differential GPS is available. Also, safety critical scenarios can be tested where pedestrians or harmful targets are involved without risking their safety during tests.Input: Interface 6 (object list)

Further information:

- M. R. Zofka, M. Essinger, T. Fleck, R. Kohlhaas and J. M. Zöllner, "The sleepwalker framework: Verification and validation of autonomous vehicles by mixed reality LiDAR stimulation," 2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), Brisbane, QLD, 2018, pp. 151-157. doi: 10.1109/SIMPAR.2018.8376285

Radar stimulator

The Radar stimulator allows the creation of virtual radar targets and therefore the testing of ADAS systems while the whole vehicle is mounted on a test bed. The radar stimulator works together with the radar target modelling software. Within the ENABLE-S3 project, functionality to stimulate targets in multiple directions has been added to the stimulator. By adding a digital stimulation module, the possible target distance has been also increased. Furthermore, the stimulator has been successfully moved from the laboratory to the more demanding environment of an industrial roller dynamometer.

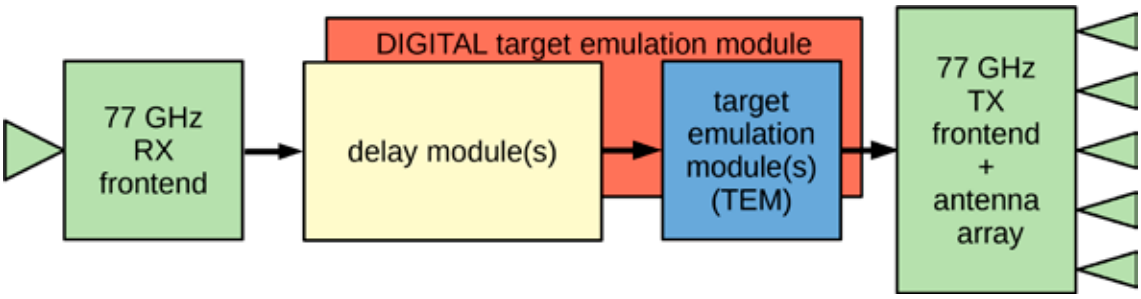


Figure 2 34: Block diagram of the radar target stimulator.

An application example is the testing of ACC functions by creating a target vehicle in front of the ego-car with variable distance and/or relative-velocity. This is a typical problem of the "highway pilot" use case.

Further readings::

- Gadringer, M. E. et al.: Radar target stimulation for automotive applications, IET Radar, Sonar & Navigation Vol. 12, Issue 10, October 2018, p. 1096-1103

- Gadringer, M. E. et al.: Virtual reality for automotive Radars, e&i – Elektrotechnik und Informationstechnik Vol. 135, Issue 4-5, August 2018, p. 335-343

- Gruber, A. et al.: Highly Scalable Radar Target Stimulator for Autonomous Driving Test Beds, Proceedings of the 14th European Radar Conference, October 2017, p. 147-150

Figure 2 35: Multi lidar sensor simulation and stimulation

Ultrasonic stimulation

Ultrasonic sensors play an important role in parking assistance systems to estimate the free space available for the car and to localize the car with respect to the surrounding obstacles. The stimulation algorithms and hardware requirements were investigated based on real sensor measurements and simulation models of the sensor and the stimulation hardware, as shown in Figure 2 35. Several stimulation architectures were analysed and evaluated. A working setup was designed in simulation by combining a model of the hardware with a model of the sensor, as shown in Figure 2 36 .

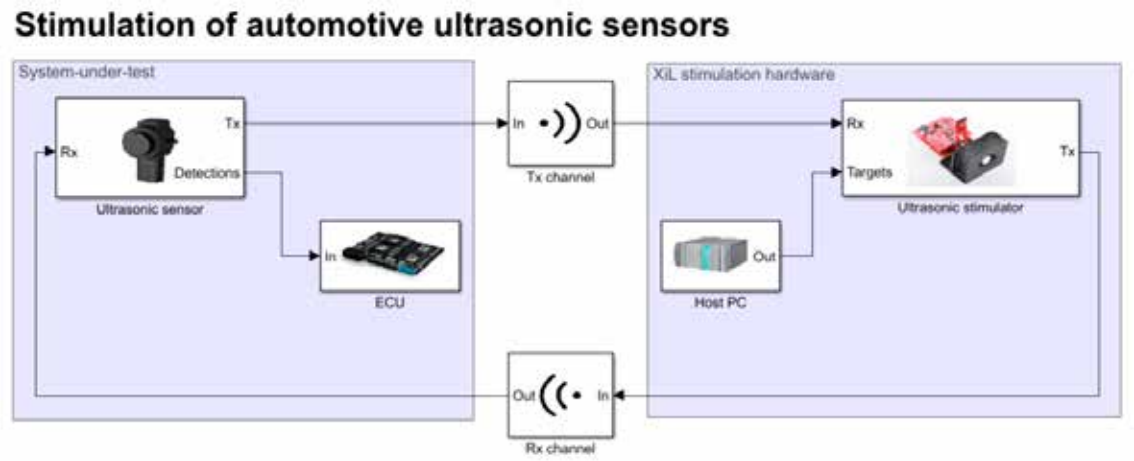


Figure 2 36: Simulation model of the ultrasonic stimulation hardware

2.5.4 Communication Channel Simulation

There is no doubt that wireless communication became an inseparable part of modern autonomous systems. Although access to sensitive data can be protected by high-level cryptography, there is a need for exploring physical aspects of communication – wave propagation, signal interferences or wireless service coverage. It is because even the most sophisticated algorithm cannot work with unavailable data, which is often critical for system's safety and security. To model those low-level level aspects, we had to develop and recreate the whole chain of communication link.

- Communication channel simulation enables detailed analysis of system behaviour under real-world conditions and events, for example:
- Signal coverage in critical infrastructure
- Impact of intentional and non-intentional signal interferences (jamming, coexisting networks)
- Different signal modulation schemes and protocols for increasing link stability, throughput and overall efficiency
- System's fail-safe behaviour in case of communication disturbances

Figure 2 37: The main parts of a wireless communication system.

Wireless communication channel simulation

GUT developed simulated counterparts of off-the-shelf transmitters and receivers (physical layer) with jamming detection capabilities and signal metrics (see Figure 2 37).

- 802.15.4 standard – used in wireless sensor networks (Zigbee, Thread and many more).
- 802.11p standard – designed for vehicular communication systems including V2X application.

Beside this, GUT developed signal interference models:

- 6 types of jammers dedicated solely for disrupting 802.15.4 and 802.11p standards,
- 10 types of radio altimeters that are known for possible interferences with coexisting systems,
- 8 types of "common" jammers including noise and constant waveform generators.

V2X channel emulator

Future connected autonomous vehicles exchange information using wireless vehicle-to-everything (V2X) communication to improve road safety and travelling convenience and to enhance the overall driving experience. Fully automated driving systems use real-time control algorithms integrated in the automated vehicle's control unit. This unit uses the information obtained via V2X communications to adapt the driving route and velocity to the current traffic scenario.

The wireless communication quality depends on the position and velocity of the vehicles as well as the surrounding environment. Wireless communication systems like 802.11p must be tested in vehicular environments in a repeatable fashion to ensure its reliability, especially in safety relevant scenarios like road crossings or traffic jams. Tests on the road, however, are typically costly, labour intensive and difficult to repeat. Vehicle-in-the loop (ViL) tests on the other hand are easily repeatable. The idea of ViL is to combine the vehicle with a virtual environment where every relevant stimulus for sensors is simulated to recreate an environment similar to a test on the road. The wireless communication channel is mimicked by a wireless channel em-

ulator. Most channel emulators, however, use rather simplistic models, like the loss of power over distance, or static delays to model the effect of the wireless communication channel. A realistic test must consider the movement of the vehicles and the update of the environment in real-time.

In the project ENABLE-S3, we developed the real-time AIT V2X radio channel emulator to properly mimic wireless communication channels. We used the geometry of the environment to model the wireless propagation channel. Although these methods can be applied in general for any scenario, in the project ENABLE-S3 we focused on a left turn scenario.

Within the project ENABLE-S3, we developed a channel model that describes a left turn scenario of a road crossing. We developed methods to update the channel characteristics in real time depending on the position of the vehicles, which was not possible before the project. The position updates of the vehicles are obtained by Virtual Test Drive by VIRES Simulationstechnik GmbH. We compared the received signal strength indicator (RSSI) and packet error rate (PER) from measurements on the road and ViL tests of the same track. The results clearly indicated that the ViL test can reproduce the PER and RSSI tests which were obtained at the test track.

The developed methods are not limited to road intersections, urban scenarios or highways. They can also be applied to mimic the wireless communication channel in industrial or railroad environments where e.g. robots communicate with each other.

Further readings::

- Hofer, M. et al.: Validation of a real-time geometry-based stochastic channel model for vehicular scenarios, in IEEE 87th Vehicular Technology Conference (VTC-Spring), Porto, Portugal, June 2018

- Hofer, M. et al.: Real-time geometry-based wireless channel emulation, IEEE Transactions on Vehicular Technology, 2019, in IEEE Transactions on Vehicular Technology, vol. 68, no. 2, pp. 1631 - 1645, February 2019 (https://ieeexplore.ieee.org/document/8584124)

2.5.5 Vehicle Controller Stimulation

Braking System Stimulation

The end-to-end performance of driver assistance systems and automated driving functions are typically dependent on the correct interactions between a chain of components, from sensing the environment until the action on the vehicle. Automated emergency braking systems (AEBS) require interactions between the ECU for the advanced driver assistance system

control function (ADAS), as well as the braking controller and hydraulic actuator. These different embedded control loops must be validated properly to guarantee correct performance of the ADAS functionality in all relevant scenarios.

During ENABLE-S3, the correct stimulation of a braking system, including its ABS/ESP control unit, was generated. This requires a real-time simulation model of the vehicle running in parallel with the real system. Outputs of the simulation are then used to provide realistic stimuli to the braking system ECU. Conversely, the output of the braking system, here the calliper pressures, are measured to provide feedback to the simulation model (Figure 2 38). The closed loop system enables the validation of the brake system controller, in combination with potential AEBS algorithms over a wide range of scenarios.

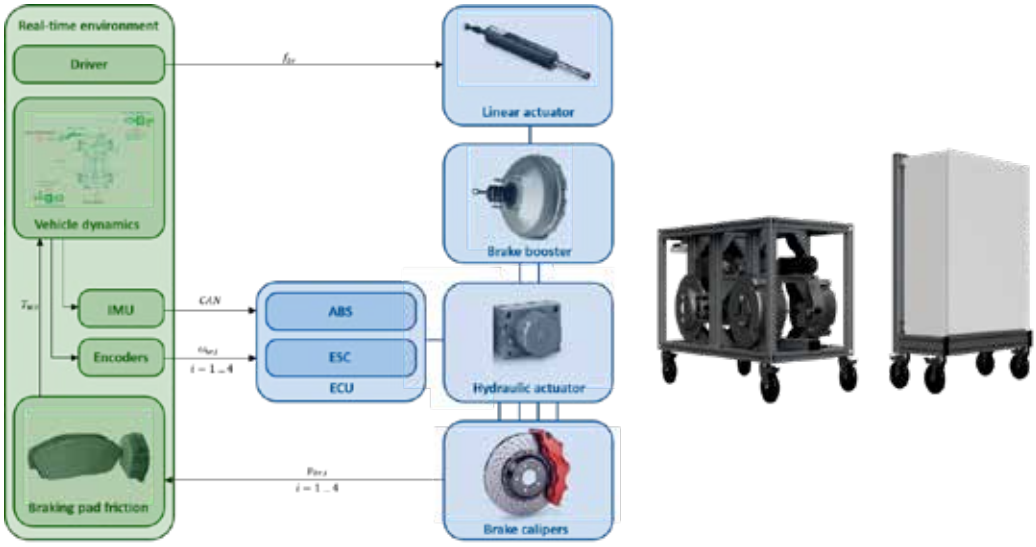


Figure 2 38: Stimulation of a braking system controller

Realistic stimulation of the ECU was studied and tested, including the generation of physical signals following the correct electrical specifications and proper communication protocols. The latter is important to enable the safe operation of the ECU, as well as to avoid triggering ECU faults due to unexpected inputs. For instance, the outputs of a simulated inertial measurement unit (acceleration and yaw rate) are transmitted over CAN-bus, which enables the ECU to estimate the velocity and rotation of the car. Additionally, it is possible to send wheel encoder signals independently for each wheel, enabling the ECU to determine whether a wheel is locked or not and to act appropriately to maintain optimal braking force. The latter has of course a direct impact on the AEBS performance. Further research steps to be investigated in a new research project include the correct simulation of the AEBS controller to enable end-to-end validation of the system.

3 IMPACT ACHIEVED IN THE APPLICATION DOMAINS

ENABLE-S3 has worked on a validation framework for automated cyber-physical systems and highly complex embedded software systems. The reference architecture and the developed validation components (bricks) have been successfully applied in the 11 industrial use cases from six different application domains. .

It has been shown that exchange among domains helps to accelerate innovation and that common validation methods can be applied in use cases of the domains.

The subsequent chapters explain how the ENABLE-S3 results on scenario-based V&V have been used in the domains and how the results will sustainably help to overcome the validation stumbling block in the introduction of highly automated systems into the market.

3.1 Impact of ENABLE-S3 in the Automotive Domain

The automotive industry still faces major challenges to bring automated driving functions on the road, demonstrated by the fact that currently there is no Level 3+ functionality available[24]. However, the required technology has been demonstrated in various prototype vehicles and first test fleets, such as by Daimler[25]. Nevertheless, regulations and homologation processes are still in work[26,27] and a commonly agreed testing methodology and supporting tool chain has yet to be defined. We believe that the ENABLE-S3 results are major contributions to this endeavour.

Complementary to research projects focusing solely on the automotive domain, such as PEGASUS (German research projects with focus on the definition of relevant scenarios) or RobustSense (ECSEL JU research project analysing sensors and sensor models for adverse conditions), ENABLE-S3 has worked on developing a set of technology bricks (toolbox), creating a tailored validation toolchain based on new and existing technologies across several domains.

Depending on the function under test and the purpose of the test, the requirements for the tool chain varies. ENABLE-S3 results support the required flexibility by providing a set of tech-

[24] *https://www.cnet.com/roadshow/news/2019-audi-a8-level-3-traffic-jam-pilot-self-driving-automation-not-for-us/*
[25] *https://www.daimler.com/innovation/autonomous-driving/special/changes.html*
[26] *http://www.unece.org/trans/themes/transtheme-its/automated-vehicles/automated-driving.html*
[27] *https://www.tuvsud.com/en/industries/mobility-and-automotive/automotive-and-oem/autonomous-driving*

nology bricks and a common reference architecture for ADAS/AD system validation along the value chain. The modular structure allows the reuse of different technology bricks (programs, products) by various OEMs, Tiers, component suppliers, academia and research institutes, tool suppliers and of course also for different applications. This modular und reusable structure enables faster and more efficient tool chain tailoring independent of the tool supplier, avoiding vendor lock-in and substantial costs for switching between products and services. The Generic Test Architecture is further aligned with other domains such as maritime, rail, aerospace, health, enabling experience and know-how exchange, especially for application independent technologies, such as co-simulation platforms and the integration of simulation and real-time components (hardware).

The modular structure can only be used in an efficient way if it supports standardized interfaces. This has also been shown in an automotive cross-use case demonstrator applying the Open Simulation Interface (OSI) to use the same modules (environment simulation and sensor model) for different applications. To ensure the sustainability of the achieved results, ENABLE-S3 contributed significantly to the standardization of OSI and two other specifications (OpenDRIVE®[28], OpenSCENARIO®[29]). As a result, standardization of developed and improved industry-defined interfaces allows to integrate existing tools at customer sites and to save significant money in the reuse of well-established tools. Moreover, the handover of results and tight cooperation with standardization ensures that the achievements are distributed in the automotive community, maintained and extended to future requirements.

Besides the setup of virtual V&V tool chains supporting the scenario-based methodology, significant advancements were made in test planning to reduce the required test time significantly and several work packages have started to analyse approaches to prove the validity of the developed tool chain.

For this reason, the research done within ENABLE-S3 will have a positive impact on the automotive domain and will pave the way for scenario-based virtual V&V and consequently the homologation of automated driving functions.

3.2 Impact of ENABLE-S3 in the Aerospace Domain

Given that we are now in the digital age, products of the aerospace industry have to fulfil the needs of the user, hence keep up with the pace of technological developments. Even though aerospace products – specifically satellites and aircrafts – have a very long lifetime, they need to show higher flexibility and better performance in their usage. Of course, these new solutions are expected to be cheaper, reliable and reach the market in less time. For satellites this

[28] *https://www.asam.net/standards/detail/opendrive/*
[29] *https://www.asam.net/standards/detail/openscenario/*

pressure is due to new space[30] (or NewSpace) and for airplanes it is due to the anticipated user experience when flying. One of the most evaluated ways to achieve better performance and lower cost is the use of COTS[31,32] (Commercial Off-The-Shelf) components. However, their applicability needs to be tested and adapted, e.g. to the space environment, and their suitability in terms of safety and security needs to be checked, e.g. in aerial transport. Both aspects have to be proven by extensive testing so sufficient confidence in its reliability and security is achieved - ENABLE-S3 has addressed these aspects.

The participation of aerospace use cases in ENABLE-S3 has allowed three main achievements. First, the use of a common reference architecture in different domains allows to divide and identify the different aspects involved in a test. Compared to the manual assessment before ENABLE-S3, it now resolves the scenarios more quickly and identifies functional areas that can be worked offline or that can be reused in the following developments. This part is done by creating databases of data recorded or created according to those collected from the real world, thus allowing these data to be ready to stimulate the system under test. Second, the development of technological bricks within the project, which can be used both in the use cases as well as others, in which special emphasis is placed on developing tools that are valid in the current application test but also in the testing of future developments.

This point drives the third achievement: the use of existing tools (e.g. TAMARIN), new methods or models from other domains (e.g. small delta to implementations in Rodin) and their adaptation to aerospace use cases. All these achievements allow to reduce the time of test and assessment (from manual to automation), the time of configuration of the test and a reduction of the cost of development. However, this use case driven project could be extended to deepen the modelling of the applications under test by testing new applications and increasing confidence in the off-line tests on the model, a test method much faster than online, and it will also allow co-simulation. Another point of improvement that can be continued is to further advance the knowledge to make the tools developed in the technological bricks more sophisticated.

In summary, the developments achieved in the aerospace use cases in ENABLE-S3 allow to test the applicability and suitability of COTS extensively. Moreover, evaluating their performance by simulating the physical environment and assessing the safe and secure placing before testing under real conditions – which allows a reduction of the cost of the test campaigns and a greater knowledge of the behaviour of the system – allows the adaption of the configuration of the test for the real test and increase the test scope by orienting the test to the critical cases. This opens the possibility of new more flexible developments, with high performance and at a lower cost than using the traditional approach of aerospace.

[30] *https://en.wikipedia.org/wiki/NewSpace*
[31] *https://www.nasa.gov/sites/default/files/atoms/files/cots.pdf*
[32] *https://wpo-altertechnology.com/cots-commercial-off-shelf/*

3.3 Impact of ENABLE-S3 in the Rail Domain

Traditionally, verification and validation V&V) in the railway domain is done manually using tests and reviews following the V-cycle given by the safety standards for this domain (CENELEC standards). In the course of the ENABLE-S3 project, a concrete example demonstrating this approach was shown with an interlocking system, where the complexity stems from the numerous different routes (possible paths for trains through a rail network) and combinations thereof which have to be tested and manually reviewed even for a comparatively small interlocking station. In the rail domain, the state of the art for verification of such a system is that test cases have to be written manually from the safety as well as functional requirements specification. At the same time, the state of the art with regards to validation is using a trace chain between hand-written customer requirements, the systems regulation rules and the system tests.

The goal in ENABLE-S3 is the development of new approaches for V&V for highly automated cyber-physical systems (ACPS). As an example, we propose the above-mentioned interlocking system as well as a hybrid ERTMS/ETCS Level 3 specification[33].

The main novelty compared to the state of the art is the newly developed approach for system validation of complex systems in the railway sector based on formal methods. This approach allows for design errors to be discovered earlier in the development process, which leads to massive savings in the V&V area of safety-critical systems. The process starts by formalizing the customer requirements into a formal domain specification, followed by an automated translation into a mathematical model, which allows for reasoning and proving properties on different levels of the (also mathematically sound) refinement.

Nowadays verification and validation is typically limited to argumentation on customer specification, implementation and test results. The great benefit of this new V&V approach is a formalized specification including a (proven) mathematical model, which allows for a higher trust in the correctness of the design and an easier validation of the design using model animation.

In ENABLE-S3 we were able to extend our verification and validation approach through the introduction of an integrated chain of methodologies and tools based on formal methods to guarantee a complete and consistent model of the domain (Interlocking respectively ERMTS/ETCS Level 3).

[33] *This is a specification for a novel ETCS approach, mixing classical interlocking techniques with block-less train-centric railway control published in 2017. See also [Hybrid ERTMS Case Study Session in ABZ conference 2018]*

During the project we were able to achieve the following sustainable impacts:

- New Process: Formal methods as a means of early V&V for safety-critical applications leading to a test environment setup time reduction of more than 70%
- Application of methodology to topology in station data for railway interlocking product leading to an increase of trust in the solution reducing the potential errors in the field
- Application of methodology to ETRMS Level 3 Specification enabling the early validation of a complex specification

3.4 Impact of ENABLE-S3 in the Maritime Domain

In the maritime domain, or Shipping, the use of simulators for testing and validation is still not common. A typical certification (type approval) will normally be done through a combination of manually test-cases, according to test standards, and sea trials where a vessel is actually tested at sea. While the automotive or aerospace industry can invest a lot of resources in one prototype, since it lays the ground work for mass production, in the maritime industry a vessel is tailored and rarely mass produced. The current practice is of course very costly and the quality or validity of the test may be questioned.

In the ENABLE-S3 project, cross domain learning has been achieved, and scenario-based V&V has been introduced to test the concept of a “Shore based bridge” (SBB). To test the SBB, six functional scenarios were considered, e.g. “Passage monitoring (sensor & traffic data in near real time)”. To do simulation-based testing, a co-simulator was established, maybe for the first time in the maritime domain, consisting of three main components;

1. Maritime Runtime Environment – eMIR platform (OFFIS, Germany)
2. HiL Ship simulator (AVL SFR, Germany)
3. Satellite communication simulator (GUT, Poland)

The benefit of the suggested ENABLE-S3 V&V concept is obvious in the maritime domain. Key Performance indicators were analysed, clearly outlying the huge potential of cost savings in combination with a scaled down, traditional type approval. By including scenarios representing actual sequences of transactions and events, bugs can be found that are often missed by other functional testing methods. Scenarios are commonly used in requirements engineering (Alexander et al. 2004[1)]) because they are easily understood by all stakeholders. A report will be made by DNV-GL – one of the leading certification companies in the maritime world - looking into how a classification company can leverage ENABLE-S3 V&V results compared to traditional approaches.

The ENABLE-S3 project, and other initiatives, have clearly made the maritime industry aware of digital twins and scenario-based V&V, but we are still in the early days. A suggested follow up action would be to use these powerful simulators into fields of enhanced safety and more environmentally friendly shipping, e.g. in meeting the International Maritime Organizations ambitious goals for reduction in Greenhouse Gas (GHG) emissions (30% by 2030, and 50% 2050) [2].

Besides the above-mentioned benefits, the project brings direct commercial benefit to NAVTOR's concept of a Shore Based Bridge. The idea is to study aspects of motion planning, monitoring and manoeuvring to shore, allowing for a crew-reduced vessel. This work has improved the cyber security in vessel-shore information exchange of info, created new tailored functionality and even kicked-off a completely new business-segment for NAVTOR; making a Fleet Centre for the Operation department at shore. This new service, named NavFleet, will enable Fleet monitoring and decision support, partly by introducing AI to utilize the data feed coming from the vessels.

In an international setting, scenario-based V&V is a must also in the maritime domain, to safely and securely test an autonomous vessel before starting to sail. And the timing is just right, the first autonomous vessel, Yara Birkeland, will start sailing, with zero-emissions, in Norwegian waters this year (2019) [3].

3.5 Impact of ENABLE-S3 in the Farming Domain

Compared to the automotive domain, the farming domain is a research area that is not that much in the news for automated and autonomous driving. Nevertheless, many technological advances have already been made in the agricultural domain that are pushing the technology forward. Examples like smart or precision farming , have provided features that focus on detection of the crops' needs and problems. These features already introduce a high level of automation and have saved millions of tons of pesticides and several working hours. On the other hand, the testing and validation of the developed technologies is still a cumbersome and time-consuming task without support from proper tools and, additionally, are heavily influenced by the availability of required resources, like e.g. farming vehicles and environmental parameters. Unfortunately, the testing and validation of many functionalities in the farming domain is very costly and dependent on external factors, like the availability of the agricultural machines, environmental (harvesting fields) and weather conditions. Especially the availability of harvesting fields (after one time harvesting, the crop is gone) is a crucial factor in the repeatability of test cases. Therefore, the partners within the use case decided to target realistic simulation environments, enabling continuous testing possibilities, without the need for

real-life vehicles or farming environments. Additionally, instead of developing new technologies, the farming use case investigated and validated technologies from other domains (e.g. automotive, maritime, aerospace) that could be applicable to the farming domain.

The ENABLE-S3 farming use case has brought together universities, research institutes and companies (SMEs and large enterprises) coming from many different areas and providing expertise from many farming related topics. The aim of the use case was to provide a testing framework consisting of multiple, partly integrated, tools enabling the validation of technologies applicable to the farming domain, varying from autonomous driving, UAVs, sensing and communication. The solutions provided in the use case are tool chains combined to solve different aspects of the identified farming problems. The first solution is a co-simulation environment targeting farming vehicle simulation (harvester, tractors and drones) in combination with sensor simulation (radar and hyperspectral).

The co-simulation environment is specialized for simulation of farming vehicles, drones and supports simulation of lidar measurements and object list detections by virtual radar. Secondly, a web-based tool suite for analysis and modelling of cyber-physical systems (CPS) for farming vehicles enables developers, inspired by the FMI standard[36], to simulate and validate scenarios of autonomous farming systems. Finally, a communication validation concept (targeting vehicle internal as well as external vehicles communications) to provide formal guarantees to improve network computations based on combination of network calculus and runtime verification and to validate the external communication based on combinatorial testing on different network parameters. These tool chains are built up in a modular and flexible way, so that new simulation models or validation technologies can easily be integrated to establish a more extensive agricultural validation environment, thereby targeting the reduction of validation time and costs.

The biggest advancement made within the farming use case is the integration of different testing and validation technologies to improve the validation effort of the overall system and not only individual technologies. Nevertheless, much integration, testing and validation is still required, especially taking the test systems towards the next level where the focus will be more on the integration of the actual hardware in the test systems (Hardware-in-the-Loop validation, HiL). This also includes real-life sensor data coming from the farming domain, thereby validating the applicability of the sensor capabilities in harsh agricultural environments.

All in all, the developments accomplished within the ENABLE-S3 farming use case provide an excellent basis for improved, time- and cost-effective validation of agricultural vehicles and scenarios. Existing simulation products and validation technologies have been extended and have created interest from industrial companies to be included in their product portfolio.

[36] *https://fmi-standards.org/*

3.6 Impact of ENABLE-S3 in the Health Domain

With the introduction of new functions and semi-autonomous movements, medical equipment is becoming increasingly complex. Verification and validation (V&V) effort of medical equipment is increasing correspondingly. In ENABLE-S3, we focused on Image Guided Therapy systems. These systems are used in hospitals for a broad range of minimal invasive procedures, like stent placement, tumour ablation or heart valve repair. The systems use semi-autonomous robotic movements, which are intended to assist the physician to manoeuvre a robot arm with an X-ray tube and X-ray detector around the patient in a user-friendly way to obtain optimal X-ray images, while avoiding collisions with patient, staff or devices in the operating room. The equipment is provided in many configurations, tailored to hospital needs. V&V of such advanced features for many configurations, procedures and workflows, requires a significant increase in test effort. At the same time, business competitiveness demands the time-to-market to be shortened.

It is investigated how to increase the efficiency of V&V by using a virtual test platform that enables virtual testing of a product instance, or part thereof, by replacing some real components with their virtual equivalents. It has been shown that a virtual test platform can reduce test costs considerably, because less physical test systems are needed. Moreover, test coverage can be increased, because a virtual test platform allows switching between configurations, enabling overnight testing of a product platform. Virtualisation also enables testing before actual hardware is available, enabling earlier feedback to development, and resolving problems in an early stage of the project. In addition, testing restrictions can be eliminated by virtualisation, in particular when 'dangerous' parts are virtualised, such as the X-ray generation.

With the virtual test platform, it is now possible to perform validation and usability tests with end users in an early phase. For example, the user interface concept of the new Philips Azurion Flexarm system[37], has been tested in a test environment with a virtual geometry, but with the real user interfaces. This resulted in several improvements in the user interaction design, which would have been much more difficult and costlier to implement at a later stage in the project.

Another instance of the virtual test platform is developed for efficient testing of the positioning software unit that controls the movements of the geometry. The positioning software can now be integrated earlier with different product configurations, using model based testing or other automated tests. This results in higher quality of the software delivered to integration testing, which significantly reduces the number of integration issues and total integration time.

[37] *https://www.philips.com/a-w/about/news/archive/standard/news/press/2019/20190117-philips-launches-azurion-with-flexarm-to-set-new-standard-for-the-future-of-image-guided-procedures.html*

Real-time X-ray imaging simulation is used in the virtual test platform, enabling a realistic user experience with sufficient image quality in a safe environment. A non real-time version of the X-ray simulation, with image quality closer to reality, is used to investigate optimal settings to reach optimal image quality with lowest possible radiation and contrast dose for specific patients and clinical procedures. By using X-ray simulation models, a much more extensive parameter space can be examined than would be possible with real patients, supported by optimization tooling.

In order to have a sustainable virtual test platform, it should be embedded in the product creation process. We investigated availability of tooling in other domains to support this. The collaboration with AVL on the applicability of Model.connect™, has provided more insight into the specific requirements for simulation test platform tooling. Currently, the virtual platform is mainly used for confidence testing. Next step is to extend the use to certification and validation.

4 ANNEX: RESEARCH RECOMMENDATIONS

In ENABLE-S3, we developed the scenario-based V&V methodology and tooling that addresses the specific aspects of highly automated systems. The technology behind highly automated systems is continuously evolving, opening many new research challenges.

Many highly automated systems are increasingly adopting machine learning (ML) and artificial intelligence (AI) to enable autonomous decision making and render applications smart. Although the use of ML and AI components gives the great promise of improving our everyday lives in many, sometimes unimaginable ways, it also opens very hard verification, validation and certification challenges in the context of safety-critical applications. The opacity of ML/AI components will require developing completely new V&V techniques and extending accordingly the existing V&V methodologies.

Modern highly automated systems are more and more often dynamically evolving after their deployment. Over-the-air updates and upgrades are becoming common in such systems, resulting in new safety and security risks. How can we ensure that updating or upgrading the system does not negatively affect one of its critical properties? How can we efficiently re-verifiy and re-certify the highly automated system after making a change? How do we perform the update/upgrade operation without jeopardizing the security properties of the system?

Within the project we evaluated different methods, tools and interfaces that have been applied in several demonstrators. The next steps are to use the experiences we made and make them applicable for users. This includes the specification of guidelines for

- Decomposition of system into subsystems. In order to build a simulation for the whole system, smaller subsystem simulation models must be created. The decomposition is relevant to improve simulation performance, stability as well as reusability.

- Interfaces for simulation units and interface design. This guideline should specify best practices for which signals should be exchanged. This is often constrained by I/O capabilities of tools or by physics. A best practice guideline would help to improve interoperability.

- Performance, simulation stability. On the one hand performance can be improved by intelligent decomposition but there are further aspects that need to be considered such as hardware, network protocol, future value prediction, etc.

Further research topics we discovered, are

- Automatic co-simulation configuration. In order to improve quality of coupling and hence improve the quality of the simulation results, co-simulation, step size and decomposition are (automatically) configured.

- Apply ViL test to multiple vehicles, obtain geometry for environment from geographical databases.

- Develop new homologation / certification processes and tools based on scenario-based verification for future product releases of ACPS

- Integrate the scenario-based V&V methods and tools into a continuous development / operation lifecycle for new ACPS products (DEVOPS paradigm from software-engineering) for continuous safe and secure product enhancements.

Various topics deserve further research to fully mature scenario-based safety validation:

- Statistical safety evidence from scenario-based verification and validation. Through real-world scenarios, better insight into the coverage of real-world systems can be provided. This means that it is better known how often certain scenario classes happen and how likely a certain parameter value is (e.g. driving 241 kph on a certain road type in France in August). But what does it mean for safety? How can we accurately predict the number of incidents? Eventually, a safety analysis can be improved by feedback from real-world monitoring of the system in use.

- Within ENABLE-S3, a great deal of consensus and common understanding was achieved. Various partners were involved in harmonization and standardization groups outside of the project consortium, e.g. SOTIF, SAE, JARA and the PEGASUS project. However, a great deal of dissemination and harmonization needs to be done to reach world-wide consensus and acceptance for the approach. Here, Europe can lead by example.

- Routine fleet monitoring for scenario mining, safety, reliability and security monitoring. Various OEMs structurally collect data from systems in operation. This can be used for scenario mining as discussed in this chapter, but also for safety surveillance, reliability monitoring and security monitoring. An integrated approach seems desirable.

- Scalability of data collection and scenario detection algorithms. With routine fleet monitoring, the amount of data and processing becomes a bottleneck during wireless transmission, storage, labelling, scenario mining and retrieval. Serious attention should be given to scalability.

- Best practices in data set sharing, including data protection measures, privacy and building trust to share.

5 ANNEX: ADDITIONAL LITERATURE

- Watzenig, Daniel, and Martin Horn, eds.: Automated Driving: Safer and More Efficient Future Driving. Springer, 2016
- www.ENABLE-S3.eu: European Initiative to Enable Validation for Highly Automated Safe and Secure Systems.
- Thorn, Eric, Shawn Kimmel, and Michelle Chaka. A Framework for Automated Driving System Testable Cases and Scenarios, No. DOT HS 812 623, 2018.
- UK Ministry of Defence (2017) Defence Standard 00-56 Issue 7 (Part 1): Safety Management Requirements for Defence Systems, Feb. 2017.

List of public deliverables

Nr.	WP	Title	Lead
D2	D1.2	D1.1.2 v1 System under test requirements and test system requirements from the automotive domain	AVL SFR
D7	D1.7	D1.2.2 v1 System under test requirements and test system requirements from the aerospace domain	AGI
D12	D1.12	D1.3.2 v1 System under test requirements and test system requirements from the rail domain	TAT
D17	D1.17	D1.4.2 v1 System under test requirements and test system requirements from the maritime domain	NAVTOR
D31	D2.1	D2.1.1 Requirements Guideline Report	TNO
D22	D1.22	D1.5.2 v1 System under test requirements and test system requirements from the health domain	PHILIPS
D27	D1.27	D1.6.2 v1 System under test requirements and test system requirements from the farming domain	TTC
D32	D2.2	D2.1.2 v1 Requirement consolidation report	TNO
D34	D2.4	D2.2.1 Metrics and measurements for evaluation report	TU/e
D37	D3.1	D3.1.1 Identification of Datasets	TNO
D45	D3.9	D3.2.2 v1 V&V Methodology	AIT
D54	D3.18	D3.4.1 Platform specification	VIF
D59	D3.23	D3.5.1 Description of requirements and of V&V criteria for component simulation and stimulation	AVL SFR
D103	D5.5	D5.1.3 Final Dissemination Plan	GUT
D109	D5.11	D5.3.1 v1 Report on IOS and RTP Contributions	VIF
D71	D4.3	D4.1.2 v1 Demonstration Report + Feedback from the automotive use cases	AVL
D76	D4.8	D4.2.2 v1 Demonstration Report + Feedback from the aerospace use cases	TAS-E
D81	D4.13	D4.3.2 v1 Demonstration Report + Feedback from the rail use case	TAT
D86	D4.18	D4.4.2 v1 Demonstration Report + Feedback from the maritime use case	NAVTOR
D91	D4.23	D4.5.2 v1 Demonstration Report + Feedback from the health use cases	PHILIPS
D96	D4.28	D4.6.2 v1 Demonstration Report + Feedback of the farming use case	TTC
D3	D1.3	D1.1.2 v2 System under test requirements and test system requirements from the automotive domain	AVL SFR
D8	D1.8	D1.2.2. v2 System under test requirements and test system requirements from the aerospace domain	AGI

Nr.	WP	Title	Lead
D18	D1.18	D1.4.2 v2 System under test requirements and test system requirements from the maritime domain	NAVTOR
D23	D1.23	D1.5.2 v2 System under test requirements and test system requirements from the health domain	PHILIPS
D28	D1.28	D1.6.2 v2 System under test requirements and test system requirements from the farming domain	TTC
D4	D1.4	D1.1.3 v1 Evaluation result of the automotive domain	AVL SFR
D9	D1.9	D1.2.3 v1 Evaluation result of the aerospace domain	AGI
D14	D1.14	D1.3.3 v1 Evaluation result of the rail domain	TAT
D19	D1.19	D1.4.3 v1 Evaluation result of the maritime domain	NAVTOR
D24	D1.24	D1.5.3 v1 Evaluation result of the health domain	PHILIPS
D29	D1.29	D1.6.3 v1 Evaluation result of the farming domain	TTC
D33	D2.3	D2.1.2 v2 Requirement consolidation report	TNO
D38	D3.2	D3.1.2 Analysis of datasets for scenario generation and sensor modelling	FZI
D39	D3.3	D3.1.3 Analysis of error effects & attack vectors	AGI
D46	D3.10	D3.2.2 v2 V&V-Methodology	AIT
D57	D3.21	D3.4.3 v1 ACPS integration platform for online/offline model coupling	VIF
D104	D5.6	D5.1.4 v1 Intermediate Communication & Dissemination Report	GUT
D110	D5.12	D5.3.1 v2 Report on IOS and RTP Contributions	VIF
D112	D5.14	D5.3.2 v1 Report on Standardisation Work	DLR
D72	D4.4	D4.1.2 v2 Demonstration Report + Feedback from the automotive use cases	AVL
D77	D4.9	D4.2.2 v2 Demonstration Report + Feedback from the aerospace use cases	TAS-E
D82	D4.14	D4.3.2 v2 Demonstration Report + Feedback from the rail use case	TAT
D87	D4.19	D4.4.2 v2 Demonstration Report + Feedback from the maritime use case	NAVTOR
D92	D4.24	D4.5.2 v2 Demonstration Report + Feedback from the health use cases	PHILIPS
D97	D4.29	D4.6.2 v2 Demonstration Report + Feedback of the farming use case	TTC
D36	D2.6	D2.3.1 Requirements tracking	AVL
D40	D3.4	D3.1.4 Reference Scenarios and benchmark for testing of ACPS and automated driving functions	PHILIPS
D41	D3.5	D3.1.5 List of recorded Datasets	TNO
D47	D3.11	D3.2.2 v3 V&V-Methodology	AIT
D58	D3.22	D3.4.3 v2 ACPS integration platform for online/offline model coupling	VIF
D120	D3.34	D3.5.1 v2 Description of requirements and of V&V criteria for component simulation and stimulation	HAGL
D119	D3.33	D3.4.1 v2 Platform specification	HAGL
D5	D1.5	D1.1.3 v2 Evaluation result of the automotive domain	AVL SFR
D10	D1.10	D1.2.3 v2 Evaluation result of the aerospace domain	AGI
D15	D1.15	D1.3.3 v2 Evaluation result of the rail domain	TAT
D20	D1.20	D1.4.3 v2 Evaluation result of the maritime domain	NAVTOR
D25	D1.25	D1.5.3 v2 Evaluation result of the health domain	PHILIPS
D30	D1.30	D1.6.3 v2 Evaluation result of the farming domain	TTC
D35	D2.5	D2.2.2 Measurements and final evaluation	TU/e
D73	D4.5	D4.1.2 v3 Demonstration Report + Feedback from the automotive use cases	AVL
D78	D4.10	D4.2.2 v3 Demonstration Report + Feedback from the aerospace use cases	TAS-E
D83	D4.15	D4.3.2 v3 Demonstration Report + Feedback from the rail use case	TAT
D88	D4.20	D4.4.2 v3 Demonstration Report + Feedback from the maritime use case	NAVTOR
D93	D4.25	D4.5.2 v3 Demonstration Report + Feedback from the health use cases	PHILIPS
D98	D4.30	D4.6.2 v3 Demonstration Report + Feedback of the farming use case	TTC
D105	D5.7	D5.1.4 v2 Final Communication & Dissemination Report	GUT
D111	D5.13	D5.3.1 v3 Report on IOS and RTP Contributions	VIF
D113	D5.15	D5.3.2 v2 Report on Standardisation Work	DLR

TABLE OF FIGURES

CONTENT OF TABLES

ENABLE-S3

VALIDATION & TESTING OF COMPLEX AUTOMATED SYSTEMS

enable-s3@avl.com

www.enable-s3.eu

AVL LIST GMBH

A-8020 Graz, Hans-List-Platz 1

www.avl.com

Acknowledgement: this project has received funding from the ecsel joint undertaking under grant agreement no 692455. This joint undertaking receives support from the european union's horizon 2020 research and innovation programme and Austria, Belgium, Czech Republic, Denmark, Finland, France, Germany, Ireland, Italy, Netherlands, Norway, Poland, Portugal, Slovakia, Spain, United Kingdom.

ECSEL Joint Undertaking

ECIA 2030

This project is member of the MOBILITY.E project cluster