SUNRISE

SAFETY ASSURANCE FRAMEWORK FOR CONNECTED, AUTOMATED MOBILITY SYSTEMS

D3.4

Report on Subspace Creation Methodology

Project short name
SUNRISE

Project full name
Safety assUraNce fRamework for connected, automated mobIlity SystEms

Horizon Research and Innovation Actions | Project No. 101069573
Call HORIZON-CL5-2021-D6-01

Funded by
the European Union

ccam-sunrise-project.eu/

Dissemination level	Public (PU) - fully open
Work package	WP3: CCAM V&V Methodology for Safety Assurance
Deliverable number	D3.4: Report on Subspace Creation Methodology
Deliverable responsible	Jobst Beckmann, ika
Status - Version	Final – V1.0
Submission date	22/08/2025
Keywords	Scenario Subspaces, Scenario Sampling, Parameter Spaces, Concretization

Authors

Role	Name
Main author	Jobst Beckmann (ika)
Contributing authors	Anders Thorsén (RISE) Erwin de Gelder (TNO) Martin Kirchengast (VIF) Bernhard Hillbrand (VIF) Thaddaeus Menzel (IDI DE) Thomas Amler (IDI DE) Fouad Hadj Selem (VEDECOM) Patrick Irvine (UOW) Eleni Daskalaki (ICCS) Emre Kaynar (AVL) José Miguel TORRES CAMARA (CAF) Marcos Nieto (VICOM) Emmanuel Arnoux (RESA)

Quality Control

	Name	Organisation	Date
Peer review 1	Stefan de Vries	IDIADA	07/07/2025
Peer review 2	Mauro da Lio	UNITN	10/07/2025

Version history

Version	Date	Author	Summary of changes
0.1	25/06/2025	Jobst Beckmann	First draft of the document.

0.2	12/08/2025	Jobst Beckmann	Revision after internal review.
0.3	21/08/2025	Jobst Beckmann	Revision after internal review.
1.0	22/08/2025	Jobst Beckmann	Final version

Legal disclaimer

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Climate, Infrastructure and Environment Executive Agency (CINEA). Neither the European Union nor the granting authority can be held responsible for them. Copyright © SUNRISE Consortium, 2025.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS AND ACRONYMS

Abbreviation	Meaning
AD	Automated Driving
ADAS	Advanced Driver Assistance Systems
ADF	Automated Driving Function
ADS	Automated Driving System
AEB	Autonomous Emergency Braking
AI	Artificial Intelligence
ALKS	Automated Lane-Keeping System
ASAM	Association for Standardization of Automation and Measuring Systems
CCAM	Connected, Cooperative, and Automated Mobility
COTSATO	COncretizing Test Scenarios and Associating Test Objectives
CT	Combinatorial Testing
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DF	Data Framework
DoE	Design of Experiments
FSM	Fuzzy Safety Model
GAN	Generative Adversarial Network
GP	Gaussian Process
GPC	Gaussian Process Classifier
GPR	Gaussian Process Regressor
IDM	Intelligent Driver Model
ISMR	In-Service Monitoring and Reporting
KDE	Kernel Density Estimation
KPI	Key Performance Indicator
LHS	Latin Hypercube Sampling

MC	Monte Carlo
MCMC	Markov Chain Monte Carlo
NATM	New Assessment/Test Method for Automated Driving
ODD	Operational Design Domain
OEM	Original Equipment Manufacturer
PCA	Principal Component Analysis
PDF	Probability Density Function
PET	Post Encroachment Time
REST	Representational State Transfer
RSS	Responsibility-Sensitive Safety
SAF	Safety Assurance Framework
SCDB	SCenario DataBase
SCM	Stochastic Cognitive Model
STPA	System-Theoretic Process Analysis
SUNRISE	Safety assUraNce fRamework for connected, automated mobIlity SystEms
SUT	System Under Test
SVM	Support Vector Machine
THW	Time Headway
TTC	Time-to-Collision
TTR	Time-to-React
UC	Use Case
V&V	Verification and Validation
V2V	Vehicle-to-Vehicle
VAE	Variational Autoencoder
WP	Work Package
XML	Extensible Markup Language

EXECUTIVE SUMMARY

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

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

After the Data Framework was queried and the logical scenarios were obtained, the concretisation step (Query & Concretise block of SAF) is needed to enable testing activities within the SUNRISE SAF (part of Execute block of SAF). This deliverable presents different concretization approaches for translating logical scenarios into concrete scenario, with the goal of creating a **Subspace Creation Methodology**, to guide testing effort into relevant subspaces of the larger parameter space. Logical scenarios, which define abstract representations of driving situations through parameter ranges and constraints, form the basis for generating systematic test case generations. The goal of this Subspace Creation Methodology is to enable a structured and efficient approach to selecting **representative test cases** from the vast space of possibilities described by logical scenarios.

A central focus of the Subspace Creation Methodology is the process of **sampling** or **selecting** specific parameter values from the logical scenarios' parameter space. This involves addressing a range of user-specific requirements, which may vary depending on the testing context or application domain. Key factors include the nature of the parameters, such as whether they are defined on a discrete or continuous scale, the availability of probability distributions for those parameters, and the characteristics of existing test instances or prior knowledge that can inform the sampling process.

This deliverable outlines a categorization of methods designed to **guide the sampling process** and improve the relevance and efficiency of test scenario generation. Among these, some are techniques that incorporate machine learning to make the **exploration** of the scenario space more intelligent and targeted. These methods have been developed specifically for the SUNRISE SAF to help **maximize coverage** of critical test cases while

minimizing the total number of concrete scenarios needed. This is especially valuable in the context of safety-critical systems, where exhaustive testing is often impractical due to the sheer size of the scenario space.

A key aspect of the Subspace Creation Methodology is the identification and prioritization of relevant **subspaces** within the overall scenario parameter space of the scenario. By determining which regions of the parameter space are most critical for the safety assessment, and which are less relevant or redundant, the Subspace Creation Methodology enables a more focused and efficient testing strategy. This reduction in scope not only improves test efficiency but also ensures that testing efforts are concentrated on areas with the highest potential safety impact.

In addition, deliverable describes metrics and methods for quantifying how well the selected concrete scenarios cover the original logical scenario space. **Coverage** is essential for evaluating the completeness and robustness of the testing process. The deliverable also discusses the **safety evaluation metric**s that must be defined and applied in order to assess the outcomes of each test case and support meaningful pass-fail decision.

.

1 INTRODUCTION

1.1 Project introduction

Safety assurance of Connected, Cooperative, and Automated Mobility (CCAM) systems is a crucial factor for their successful adoption in society, yet it remains a significant challenge. CCAM systems need to demonstrate reliability in all driving scenarios, requiring robust safety argumentation. It is acknowledged that for higher levels of automation, the validation of these systems by means of real test-drives would be infeasible. In consequence, a carefully designed mixture of physical and virtual testing has emerged as a promising approach, with the virtual part bearing more significant weight for cost efficiency reasons.

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

The initiatives mentioned above, provide new approaches to CCAM validation, and many expert groups formed by different stakeholders, are already working on CCAM systems' testing and quality assurance. Nevertheless, the lack of a common European validation framework and homogeneity regarding validation procedures to ensure safety of these complex systems, hampers the safe and large-scale deployment of CCAM solutions. In this landscape, the role of standards is paramount in establishing common ground and providing technical guidance. However, standardising the entire pipeline of CCAM validation and assurance is in its infancy, as many of the standards are under development or have been very recently published and still need time to be synchronised and established as common practice.

Scenario Databases (SCDBs) are another issue tackled by several initiatives and projects, that generally tends to silo solutions. A clear, concrete approach should be used (at least at European level), dealing with scenarios of any possible variations, including the creation, editing, parameterisation, storing, exporting, importing, etc. in a universally agreed manner.

Furthermore, validation methods and testing procedures still lack appropriate safety assessment criteria to build a robust safety case. These must be set and be valid for the whole parameter space of scenarios. Another level of complexity is added, due to regional differences in traffic rules, signs, actors, and situations.

Evolving from the achievements obtained in HEADSTART and taking other project initiatives as a baseline, it becomes necessary to move to the next level in the development and demonstration of a commonly accepted **Safety Assurance Framework** (**SAF**) for the safety validation of CCAM systems, including a broad portfolio of Use Cases (UCs) and comprehensive test and validation tools. This will be done in **SUNRISE**, which stands for **S**afety ass**U**ra**N**ce f**R**amework for connected, automated mob**I**lity **S**yst**E**ms.

The SAF is the main product of the SUNRISE project. As the following figure indicates, it takes a central role, fulfilling the needs of different automotive stakeholders that all have their own interests in using it.

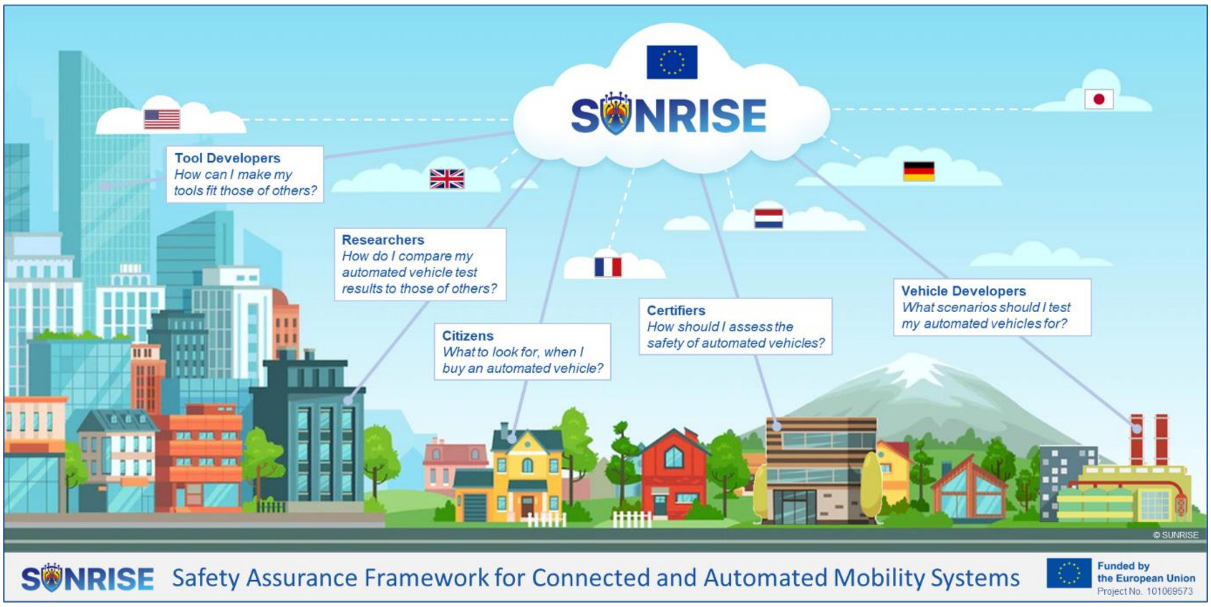

SUNRISE
Tool Developers
How can I make my tools fit those of others?
Researchers
How do I compare my automated vehicle test results to those of others?
Citizens
What to look for, when I buy an automated vehicle?
Certifiers
How should I assess the safety of automated vehicles?
Vehicle Developers
What scenarios should I test my automated vehicles for?
© SUNRISE
SUNRISE Safety Assurance Framework for Connected and Automated Mobility Systems
Funded by the European Union
Project No. 101069573

Figure 1: Safety Assurance Framework stakeholders

The **overall objective** of the SUNRISE project is to accelerate the safe deployment of innovative CCAM technologies and systems for passengers and goods by creating demonstrable and positive impact towards safety, specifically the EU's long-term goal of moving close to zero fatalities and serious injuries by 2050 (Vision Zero), and the resilience of (road) transport systems. The project aims to achieve this objective by providing a SAF consisting of three main components: a Method, a Toolchain, and a Data Framework. The **Method** is established to support the SAF safety argumentation, and includes procedures for scenario selection, sub-space creation, dynamic allocation to test instances, and a variety of metrics and rating procedures. The **Toolchain** contains a set of tools for safety assessment of CCAM systems, including approaches for virtual, hybrid, and physical testing. The **Data Framework** provides online access, connection and harmonization of external Scenario Databases (SCDBs), allowing its users to perform query-based extraction of safety relevant scenarios, allocation of selected scenarios to a variety of test environments, and generation of the test results. The SAF will be put to the test by a series of **Use Cases demonstrations**, designed to identify and solve possible errors, gaps, and improvements to the underlying methods, tools, and data.

Following a common approach will be crucial for present and future activities regarding the testing and validation of CCAM systems, allowing to obtain results in a standardised way, to improve analysis and comparability, hence maximising the societal impact of the introduction of CCAM systems.

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

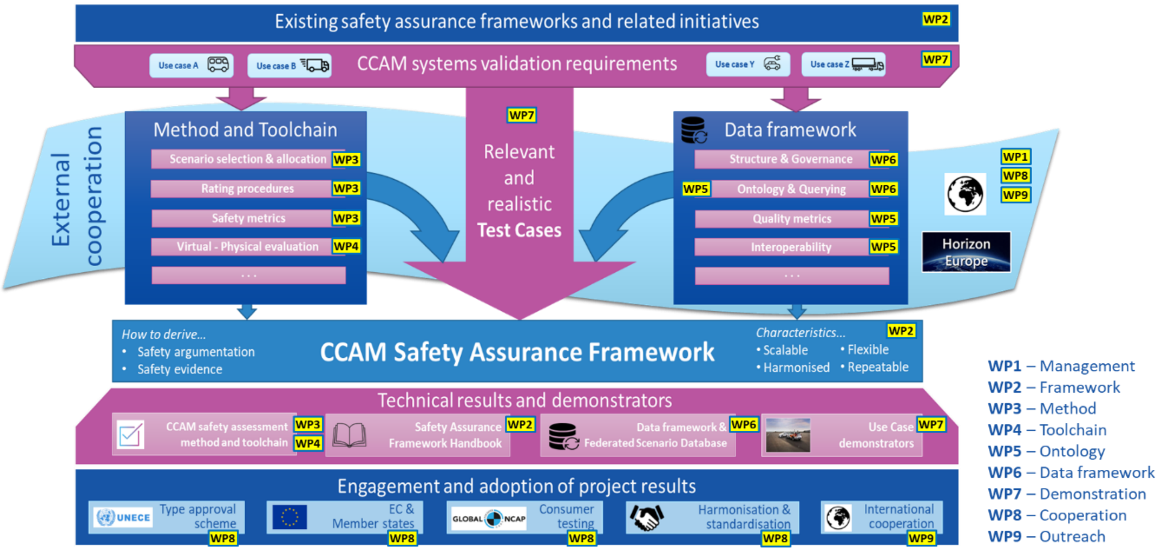

Existing safety assurance frameworks and related initiatives
WP2
Use case A
Use case B
CCAM systems validation requirements
Use case Y
Use case Z
WP7
External cooperation
Method and Toolchain
Scenario selection & allocation WP3
Rating procedures WP3
Safety metrics WP3
Virtual - Physical evaluation WP4
. . .
WP7
Relevant and realistic Test Cases
Data framework
Structure & Governance WP6
WP5 Ontology & Querying WP6
Quality metrics WP5
Interoperability WP5
. . .
WP1
WP8
WP9
Horizon Europe
How to derive...
• Safety argumentation
• Safety evidence
CCAM Safety Assurance Framework
Characteristics...
• Scalable
• Flexible
• Harmonised
• Repeatable
WP2
Technical results and demonstrators
CCAM safety assessment method and toolchain WP3 WP4
Safety Assurance Framework Handbook WP2
Data framework & Federated Scenario Database WP6
Use Case demonstrators WP7
Engagement and adoption of project results
UNECE Type approval scheme WP8
EC & Member states WP8
GLOBAL NCAP Consumer testing WP8
Harmonisation & standardisation WP8
International cooperation WP9
WP1 – Management
WP2 – Framework
WP3 – Method
WP4 – Toolchain
WP5 – Ontology
WP6 – Data framework
WP7 – Demonstration
WP8 – Cooperation
WP9 – Outreach

Figure 2: Workplan of the SUNRISE Project

1.2 Purpose of deliverable

This deliverable outlines the Subspace Creation Methodology for deriving concrete test scenarios from logical scenarios extracted via the SUNRISE Data Framework. It serves as a bridge between the query process developed within the SUNRISE project, first detailed in Deliverable D6.1, and the allocation strategy for mapping test scenarios to test instances, as described in Deliverable D3.3.

The document provides an overview of sampling and selection approaches, designed not only for SUNRISE-specific use cases but also adaptable to external applications. These include approaches from literature, as well as solutions that have been developed within the SUNRISE project. The different approaches are structured and categorized in a framework. The purpose of this framework is to provide guidelines for selecting the most effective approach under various conditions, including differing data source availability. Additionally, strategies for structuring the extensive parameter space of logical scenarios into smaller, more manageable sub-spaces tailored to testing CCAM systems using the sampling and selection approaches are presented. Using smart sapling algorithms to focus testing in scenario sub-spaces is the focus of this deliverable.

1.3 Intended audience

The primary audience includes all stakeholders, both internally and externally, who require a methodology to generate test scenarios for the safety assurance of CCAM systems. Specifically, the contents of this deliverable should be used within the SUNRISE project to help determine the methodology to generate concrete scenarios for the project's use cases. External stakeholders can use this deliverable to guide their application of the Concretization step within the Query & Concretize block of the SUNRISE SAF.

1.4 Deliverable structure and relation to other parts of the project

The work presented in this deliverable is intricately linked to the activities in other technical work packages, forming a cohesive part of the overall project's Safety Assurance Framework (SAF). Its connection to WP4 is evident, as the methodology developed here, serves as the foundation for generating concrete test scenarios critical to WP4's objectives. Similarly, the logical scenarios produced extracted by the SUNRISE Data Framework in WP6 seamlessly feed into this Subspace Creation Methodology, highlighting the collaborative nature of these efforts.

In WP5, the ontology and metrics play an important role, providing essential elements that are integrated into and utilized by the methodology. Finally, the real-world applicability of this work is showcased in WP7, where the methodology is demonstrated in practice.

The sections that follow delve deeper into these relationships, providing a comprehensive view of how this deliverable aligns with and supports the goals of the different work packages and tasks across the project.

1.4.1 Relation to WP2

As mentioned above, this deliverable is intricately linked with the project's Safety Assurance Framework. WP2 concerns the developments of this framework and therefore dictates how all other work packages will interact with it.

In particular, the methodology of subspace creation interacts with the Safety Assurance Framework as part of the Query and & Concretise block, visualised in Figure 3. Work package 2 has defined the workflow for which subspace creation features. From the SAF, we know that scenarios can be retrieved from individual scenario databases, via the SUNRISE Data Framework (DF), in both logical and concrete levels of abstraction.

All parameters of logical scenarios are defined using value ranges (including infinite such as curvature radius covering curves and straights), allowing for an infinite number of concrete scenarios to be derived from a single logical scenario. The next step is to concretise these parameter ranges into specific values and to combine these concrete scenarios with the test objectives. Methods of selecting and concretising scenarios are not covered within the scope of WP2 and rather are rather explored here and in related work packages.

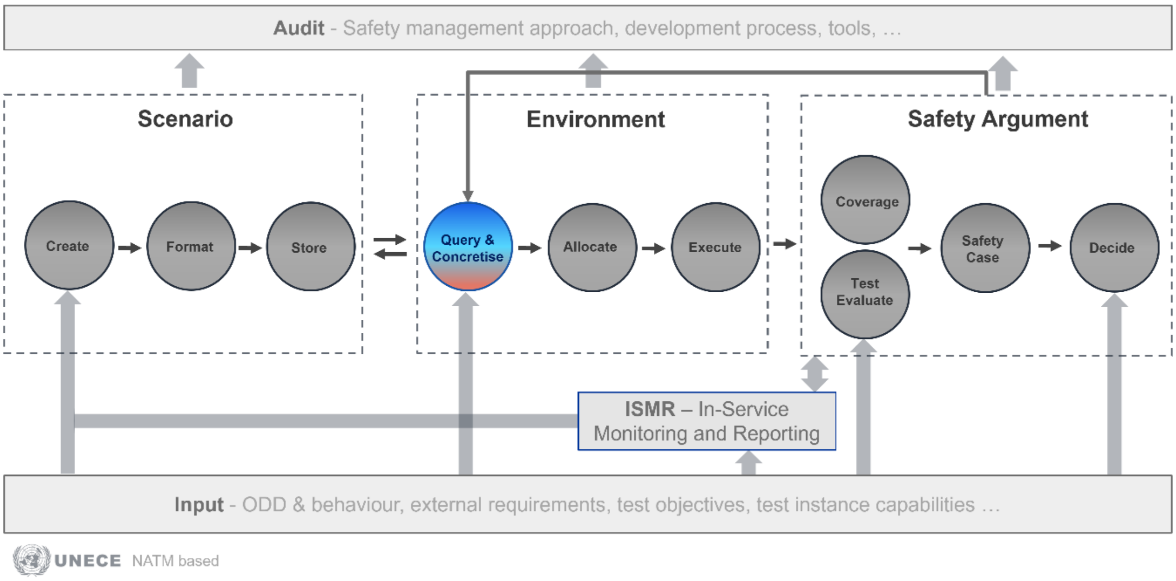

Audit - Safety management approach, development process, tools, …
Scenario
Create
Format
Store
Environment
Query & Concretise
Allocate
Execute
Safety Argument
Coverage
Test Evaluate
Safety Case
Decide
ISMR – In-Service Monitoring and Reporting
Input - ODD & behaviour, external requirements, test objectives,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

Figure 3: SUNRISE SAF, highlighting the position of the 'Query and Concretise' block within the process.

1.4.2 Relation to WP3

The deliverable D3.4 plays a key role within the structure of WP3, as it describes the concretization step in the Query & Concretize block of the SUNRISE SAF, in addition to giving information on other blocks, like Coverage, as well. The deliverable builds upon the requirements defined in deliverable D3.2 [1]. Additionally, the content of the deliverable D3.4 is interconnected with that of deliverable D3.5 [2], which describes the validation of test runs and deliverable D3.3 [3], which takes the concrete scenarios sampled or selected by the methods described in this deliverable as an input.

1.4.3 Relation to WP4

Although the methods listed in this deliverable do not influence the development in the WP4 tasks, this work is an important prerequisite for the 'Execute' block of the SUNRISE SAF. It ensures a meaningful selection and combination of parameters and thus reduces the number of tests required, resulting in less time and cost for the execution of the tests.

1.4.4 Relation to WP5

The efforts within WP5 give support to the efforts within this deliverable by providing definitions for formats and ontologies. Especially important are the definition of metrics within deliverable D5.3 [44] that are used for the methodology as well.

1.4.5 Relation to WP6

The concretization process described within this deliverable is the step directly succeeding the querying as defined in WP6. The output of the querying process are scenarios that can either be tested directly or must be concretized in a process that is described in this deliverable. As such, it is very important to consider the formats, as well as the structure of the querying process and the related tools developed within WP6.

1.4.6 Relation to WP7

Within WP7, the SUNRISE SAF is put to the test by evaluating the safety of different CCAM systems in different use cases. The resulting methodology of this deliverable was used in WP7 to enable the concretization of the use case scenarios to enable simulation, proving ground, and hybrid testing. A description of the use cases is given in deliverable D7.2 [4].

2 SCENARIOS AND PARAMETER SPACES

This chapter establishes the foundation for the methodology Subspace Creation Methodology detailed in the subsequent chapters. To achieve this, it begins by defining the key terms and concepts essential to the discussion, such as logical and concrete scenarios, as well as scenario parameter spaces. This gives the necessary background for the Subspace Creation Methodology that follows.

Most of the groundwork for concepts and terminology has been provided by the deliverable D3.2 [1]. The deliverable D3.2 does also discuss requirements related to scenarios, some of which are very relevant for this deliverable as well. In this chapter the relevant content from D3.2 is highlighted and expanded to form the basis for further content in this deliverable.

2.1 Scenario Definition

A brief definition of a scenario and its abstraction level is required, as this is the foundational concept surrounding the methods and approaches discussed in this deliverable. For a more detailed definition one should refer to deliverable D3.2 [1], as well as the commonly accepted standard ISO 34501. A scenario, as defined by deliverable D3.2 [1] is a description of a temporal and spatial traffic constellation, that includes the specific set of conditions or and events that an ADS may encounter during its operation.

Another definition is provided by Ulbrich et al. (2015), who define a scenario as a sequence of scenes that capture the state of an environment over time. A scene, as defined by ISO 34501, is a snapshot of all entities. These scenes may include actions, events, conditions, or goals that shape their temporal evolution. Scenarios offer a structured and deterministic framework for representing subjective situations, such as those a driver might encounter. The outcome of a scenario, however, remains undetermined, influenced by the interactions of various actors involved.

Scenarios can be classified based on their level of abstraction into four main types: functional, abstract, logical, and concrete. A description of different scenario abstractions is provided by [5]. Functional scenarios, being the most abstract, provide high-level descriptions of events and conditions in natural language. The abstract scenario is a more formalized description of a functional scenario based on an ontology. Abstract scenarios are further refined into logical scenarios, where parameters and their ranges are specified, such as the initial speed of an ego vehicle. At the most detailed level, concrete scenarios assign precise values to these parameters, offering a complete and specific representation of the scenario.

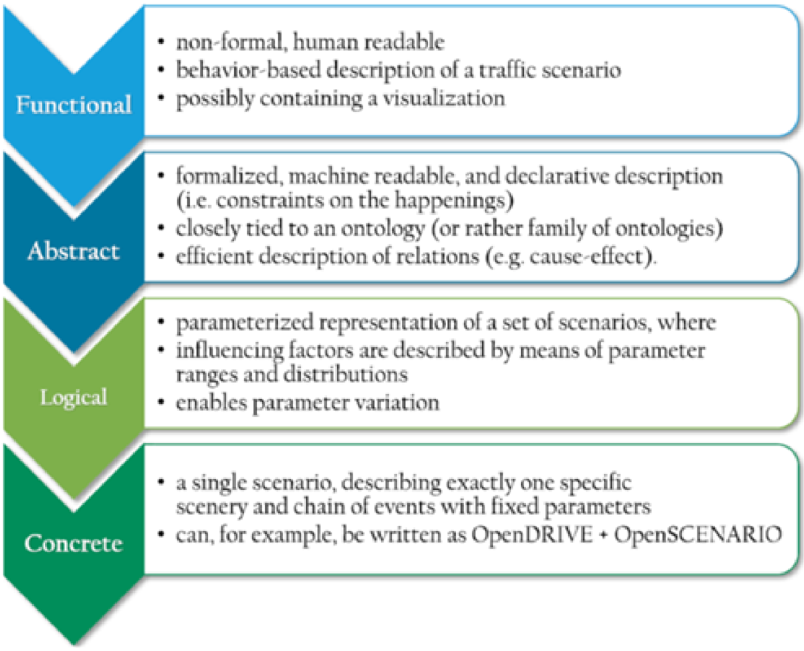

Functional
• non-formal, human readable
• behavior-based description of a traffic scenario
• possibly containing a visualization
Abstract
• formalized, machine readable, and declarative description (i.e. constraints on the happenings)
• closely tied to an ontology (or rather family of ontologies)
• efficient description of relations (e.g. cause-effect).
Logical
• parameterized representation of a set of scenarios, where
• influencing factors are described by means of parameter ranges and distributions
• enables parameter variation
Concrete
• a single scenario, describing exactly one specific scenery and chain of events with fixed parameters
• can, for example, be written as OpenDRIVE + OpenSCENARIO

Figure 4: Different scenario abstraction levels.

Beyond levels of abstraction, scenarios can also be categorized by their content. One common approach is the six-layer model proposed by [6], which organizes scenario elements into distinct categories:

1. **Road Network and Traffic Guidance Objects:** This layer describes static infrastructure like lanes and traffic signs.

2. **Roadside Structures:** Includes objects such as buildings and vegetation adjacent to the road.

3. **Temporary Modifications:** Captures changes to layers 1 and 2, such as construction zones or temporary barriers.

4. **Dynamic Objects:** Represents moving entities, such as vehicles and pedestrians.

5. **Environmental Conditions:** Covers factors like weather, lighting, and visibility.

6. **Digital Information:** Encompasses elements such as navigation data and vehicle-to-infrastructure communication.

For scenarios to be operational, particularly in applications like traffic simulations, they must be systematically defined within these layers, ensuring consistency and interoperability. This is often achieved through the use of ontologies that capture the relationships between scenario elements. Additionally, a standard format is necessary to represent the scenario content in a way that is both human-readable and machine-interpretable.

Industry standards established by the Association for Standardization of Automation and Measuring Systems (ASAM) provide essential tools for this purpose. Key among these are OpenSCENARIO, which handles dynamic content such as the behaviour of vehicles and pedestrians, and OpenDRIVE, which describes static road network configurations. Together, these standards enable the creation of robust and reusable scenarios for a wide range of applications.

2.2 Parameter Space Definition

The parameter space, as defined by deliverable D3.2 [1], refers to the entire range and combinations of these scenario parameters in a logical scenario. The scenario parameters create together form a multidimensional representation of all relevant variables that characterize a driving scenario. These parameters encapsulate a wide range of aspects, typically categorized into the following domains:

- Environmental parameters: weather conditions (e.g., rain, fog, snow), lighting (e.g., day, night, dusk), and road surface conditions.
- Traffic parameters: presence and behaviour of other road users (e.g., cars, pedestrians, cyclists) or objects, traffic density, and traffic rule compliance.
- Road infrastructure parameters: road type (e.g., highway, urban street), geometry (e.g., curves, slopes), lane markings, intersections, and signage.
- Vehicle dynamics parameters: speed, acceleration, braking capability, and sensor configurations of the ego vehicle.
- Interaction parameters: temporal and spatial gaps between vehicles, merging behaviour, and cooperative communication patterns in connected environments.

The parameter space must be sufficiently comprehensive to capture the variability and complexity of real-world scenarios, while also being tractable for simulation and analysis. To this end, ranges and discretizations are typically defined for each parameter, either through expert knowledge, empirical data, or regulatory requirements.

An effectively defined parameter space allows for the generation of a wide spectrum of test cases, ranging from common to edge-case scenarios. It also forms the foundation for scenario-based testing methodologies, such as those discussed in this deliverable.

2.3 Selection and Sampling

In academic publications and discussions, "sampling" and "selection" are often used interchangeably, though they differ significantly. This confusion arises partly from the frequent association of both concepts with creating scenarios or datasets for autonomous driving system verification, often by testing or falsification. However, methodologically, there is an essential distinction: "sampling" specifically refers to generating simulated data based on probabilistic models, while "selection" involves choosing real scenarios observed in an existing database. This distinction can significantly impact verification results, as selected scenarios reflect real situations, while sampled scenarios rely on probabilistic assumptions.

3 GENERAL CONCRETIZATION APPROACHES

It is not possible to identify a singular approach for subspace creation and the underlying selection and sampling methods that provides a consistent methodology. Instead, this deliverable presents a variety of different approaches applicable in various situations. To facilitate understanding, these approaches are structured to highlight their commonalities and differences. In this chapter, a unifying concretization categorization framework is introduced that organizes these methods. Additionally, various approaches from literature are presented that align with our categorization. The following Chapter 4 will shift focus to the specific approaches developed in SUNRISE aimed at addressing the concretization and subspace creation problems discussed in Chapter 4.

3.1 Scenario Concretization

The main goal of any approach is the creation of concrete scenarios, since these are the scenarios required for testing, both in simulation, and proving ground test instances. The scenario databases are however frequently storing logical scenarios. This is also sensible, since every logical scenario can include an infinite number of concrete scenarios. The process of deriving a finite number of concrete scenarios from a logical scenario is called scenario concretization. This process also takes an important place within the SUNRISE SAF, having, together with the querying process, a block, the Query and Concretise block, dedicated to it. To conduct the scenario concretization process, sampling methods are required. The goal of these methods is to determine an appropriate number of samples, or parameter combinations, as well as the combinations themselves to test the CCAM system.

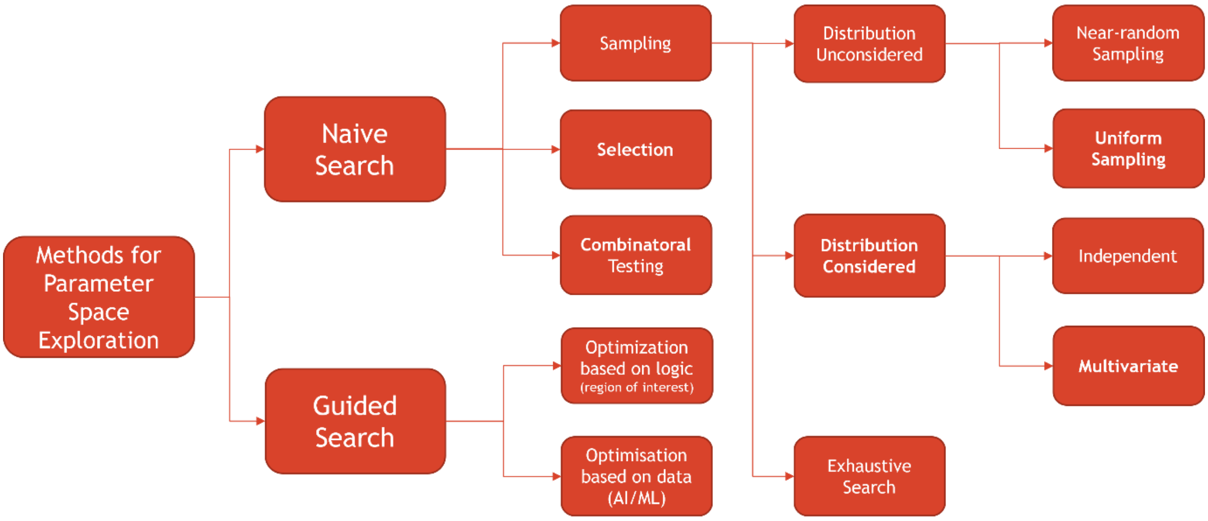


Figure 5: Structure of the Concretization Categorisation Framework

The overall structure of the categorization framework is described in Figure 5. On a high level, we distinguish between two main categories, Naïve Search and Guided Search. The high-level structure of the framework as it is presented in this chapter is based on [7], with further adaptations to fit the SUNRISE SAF. The categories of the framework are further explained in the next sections.

3.2 Naïve Search

For a use case with a scenario space of lower dimensionality, it is feasible to use a naïve approach for scenario exploration by randomly or systematically searching over the scenario space. Since all samples are mutually independent, they can be processed in parallel to reduce computing time. However, as the logical scenarios become more complex and problem dimension increases using naïve methods can become infeasible.

3.2.1 Sampling with distributions unconsidered

In many real-world scenarios, we often need to generate samples without having any prior knowledge of the underlying distribution. This is common in early-stage modelling, simulations, or exploratory analyses where the input space is defined, but its probabilistic structure is unknown. In such cases, sampling methods that do not rely on a predefined distribution become particularly valuable. Two widely used techniques in this category are random sampling and Latin hypercube sampling.

Random sampling is perhaps the most intuitive approach. It involves selecting points uniformly at random from the input space, treating each dimension independently and without bias. This method is easy to implement and works well in simple or low-dimensional settings. However, as dimensionality increases, random sampling can lead to uneven coverage, some areas may be oversampled, while others are left sparse, potentially reducing the reliability of downstream analyses, since randomness can miss certain rare characteristics [8]. A solution can be to use stratified sampling instead.

One sampling approach to reducing the variance is Latin hypercube sampling (LHS), which offers a more structured alternative. Instead of drawing points entirely at random, LHS divides the range of each variable into equal intervals and ensures that each interval is sampled exactly once. This stratified strategy leads to better coverage of the input space, making it especially useful in high-dimensional problems or when computational efficiency is a concern. Despite its more complex setup, LHS provides a significant improvement in the variance of the sample set without assuming any prior knowledge of distributions [9].

Various other sampling methods are available, each suited to specific needs, such as Monte Carlo (MC) methods [10], Metropolis-Hastings [11], Gibbs sampling [12], and Markov Chain Monte Carlo (MCMC) [13]. These techniques are widely used to generate test or evaluation scenarios based on probabilistic models.

3.2.2 Sampling with distributions considered

Typically, multiple parameters are considered for a (test) scenario. In addition, at least some of the values of these parameters are dependent. Thus, multivariate distributions are needed to model the probability density that is underlying the parameter values of the real-world scenarios.

To model multivariate distributions, two distinct approaches can be considered: parametric distributions and non-empirical distributions. With parametric distributions, a functional form is chosen and parameters of these distributions (not to be confused with the earlier-mentioned scenario parameters) are fitted to the data. With non-parametric distributions, there is no functional analytical expression for the probabilities chosen. The benefit of this approach is that the shape of the distribution automatically adapts to the data. This goes at a cost: fitting a non-parametric distribution typically requires more data than fitting a parametric distribution. Thus, if a functional form of the specific distribution can be justified, this should be preferred. In the case of the scenario parameters, however, there is generally no justification for assuming a particular functional form of the distribution.

In [14] the use of Kernel Density Estimation (KDE) to fit the scenario distribution is proposed. If a set of observed d-dimensional scenario parameters is denoted by $\{x_i\}_{i=1}^{n}$, then the estimated probability density function with KDE is:

$$f(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right).$$

Here, $K(\cdot)$ is the so-called kernel function, and $h > 0$ is the bandwidth. Intuitively, KDE has the effect of smoothing out each data point into a bump whose shape is determined by $K(\cdot)$. Figure 6 illustrates a KDE in one dimension. Here, the data points are represented by the black vertical lines, which are then smoothed by the kernel illustrated by the red curves. The final result is obtained by adding the individual red curves, resulting in the blue line. Note that here, use is made of the Gaussian kernel, given by

$$K(u) = \frac{1}{(2\pi)^{\frac{d}{2}}} \exp\left\{-\frac{1}{2}\|u\|_2^2\right\},$$

where $\|u\|_2^2 = u^{\mathrm{T}}u$ denotes the squared 2-norm of u.

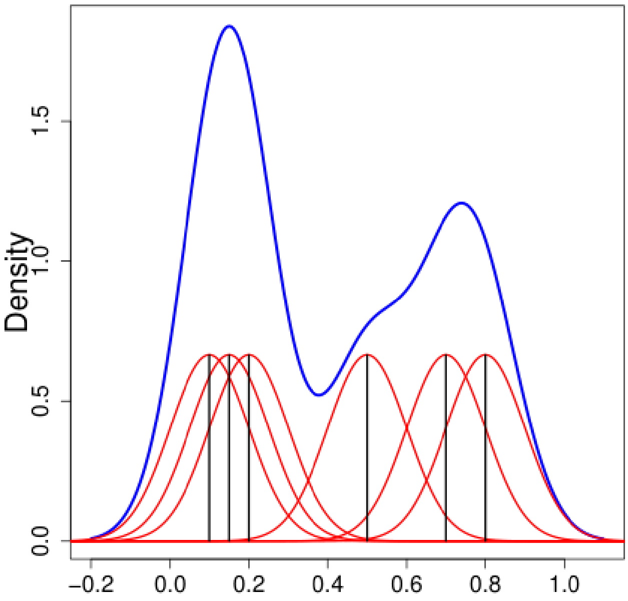


Figure 6: Illustration of Kernel Density Estimation in one dimension. Figure taken from [15].

Sampling from a KDE is straightforward: two random numbers are drawn, one to choose a random generator (ranging from 1 to n), and one random number from the chosen kernel. KDE also allows conditional sampling, i.e., sampling parameter values while fixing one or more parameters to a particular value [16].

Typically, the choice of the kernel function is less important than the choice of the bandwidth. A larger bandwidth results in a smoother result. Choosing h too large may result in loss of details. Methods to determine the bandwidth range from simple reference rules like Silverman's rule of thumb [17] to more elaborate methods, such as cross-validation methods and plug-in methods [18]. Note that often the same amount of smoothing is applied in every direction; it is also possible to use a bandwidth matrix H to apply different smoothings, in which case $K\left(\frac{x-x_i}{h}\right)$ becomes $K\left(H^{-1}(x-x_i)\right)$.

Other non-parametric probability density estimation includes the use of normalizing flows [19]. Compared to KDE, normalizing flows scale better with the dimensionality of the data, which makes them a popular approach when many dimensions are considered. The main idea of a normalizing flow is to transform a known base density, such as a Gaussian distribution, to the density of the underlying data using some invertible transformation. Provided that a base density is chosen from which it is straightforward to sample, sampling values from a density estimated using normalizing flows is straightforward too: a number is generated from the base density and then transformed using the transformation towards the original parameter space. A first study on the use of normalizing flows for estimating distributions of scenario parameters is performed in [20] and [21]. First results indicate that normalizing flows can be utilized to estimate probability density functions. It may require more computational resources and tweaking of the hyperparameters, but initial results suggests that normalizing flows are less prone to the curse of dimensionality. Thus, normalizing flows may be particularly useful, compared to KDE, if the number of scenario parameters is increasing.

3.2.3 Exhaustive Search

Exhaustive search (also denoted exhaustive testing) is a straightforward method, for which full coverage can be guaranteed, that is feasible when the scenario space is fully discrete and reasonable reasonably small [22]. One approach for exhaustive search is to discretise the continuous parameters, followed by performing a full-scale grid search examining every existing test case in the scenario space. Alternatively, for the case scenarios that are available from a scenario database, all scenarios derived from the database are checked.

3.2.4 Selection

In academic publications and discussions, "sampling" and "selection" are often used interchangeably, though they differ significantly. This confusion arises partly from the frequent association of both concepts with creating scenarios or datasets for autonomous driving system verification, often by testing or falsification. However, methodologically, there is an essential distinction: "sampling" specifically refers to generating simulated data based on probabilistic models, while "selection" involves choosing real scenarios observed in an existing database. This distinction can significantly impact verification results, if the selected scenarios reflect real situations, while sampled scenarios generally rely on probabilistic assumptions.

Selection is important, since it can be impractical to process every concrete scenario collected from scenario databases containing concrete scenarios due to the sheer volume, which makes handling and processing labour-intensive and complex. Thus, it becomes essential to extract a manageable sample that is feasible in terms of time and budget. Hence reducing the size of this real-world database while retaining the breadth of information it contains and accounting for inherent redundancies is a main issue. This raises the crucial question of selecting reduced subsets out of a set of concrete scenarios from the same logical scenario that maintain all essential information from the total population with fewer elements to facilitate subsequent testing. Indeed, in the scenario-based approach to evaluating automated vehicles, careful concrete scenario selection is essential to determine which specific scenarios will undergo testing. Choosing the right scenarios is fundamental for a comprehensive and effective evaluation of automated vehicles' performance and safety.

Scenario selection, particularly in the context of driving, is less commonly used and less explored in the literature than sampling approaches that have been discussed, as it generally arises after collecting a large database and requires substantial investment for extracting meaningful real cases. For instance, [23] focus on identifying high-risk scenarios for cooperative and automated vehicles using simulation-based criticality assessments, directing testing efforts towards significant cases. [24] propose a relevance-based filtering method to reduce the number of test scenarios while ensuring comprehensive coverage of critical cases. [25] utilize clustering techniques to group similar scenarios, allowing the selection of a representative subset for continuous validation of highly automated driving systems, maintaining diversity within a reduced dataset. These approaches collectively aim to optimize selection test examples while preserving robust coverage of essential driving scenarios.

Though complex and data-intensive, selecting real scenarios offers the advantage of providing real driving situations with fully recorded data, enabling more realistic validation and a deeper understanding of autonomous vehicle behaviour. Additionally, scenario selection allows working with complete multidimensional data, enabling comprehensive analysis of all relevant variables in real-world driving.

The recent approach proposed in [26] combines these aspects: selecting real occurrences from samples using an equiprobable partition of the continuous variable space while leveraging this same partition for stratified simulations. Inspired by the concept of Latin Hypercube Sampling, this method differs by not requiring an independence assumption between variables and adapts even with numerous variables. This improvement is achieved by efficiently selecting real, simulated, or mixed sub-populations in a statistically representative manner, even for a large number of variables. It thus offers an integrated solution to optimize both real data selection and simulation, enabling more efficient evaluation of autonomous driving systems.

3.2.5 Combinatorial Testing

Combinatorial testing (CT) focusses on identifying failures triggered only by combinations of inputs [22]. For an ADS or a particular AD function, it can be used to find the unknown combinations of influential factors that may cause failure of the system. Core in CT is to generate a minimum set of test cases, a covering array, that satisfies N-wise coverage. The covering array stores the testing configurations where each row of the array can be regarded as a set of parameter values for a specific test. Computation cost can be significantly reduced, and test efficiency increased with a good covering array. However, to find a minimum covering array is an NP-hard problem. Here, NP stands for nondeterministic polynomial time, meaning that if someone gives you a proposed solution, you can check its correctness quickly (in polynomial time). NP-hard means that the problem is at least as hard as the hardest problems in NP; no known algorithm can guarantee an optimal solution quickly for all cases, and it is widely believed none exists. In practical terms, the time required to find the minimum covering array grows explosively as the number of parameters and values increases.

As N-wise coverage can only be defined on discrete parameter space, continues variables in the parameter space of logical scenarios must be handled. One approach is to discretize them assigning specific values to them. Another approach is to perform the CT in two steps. In the first step, CT is done over all discrete parameters. Then optimization is conducted through simulations on all continuous parameters for each combination of discrete parameters. Other approaches to generate covering arrays besides on focusing on minimizing the number of test cases is also possible. There are examples where the generated covering matrix fulfils not only the N-wise coverage but also maximizes the overall complexity of all the scenarios.

3.3 Guided search

If little is known about the parameter space to be explored, a guided search can be an effective strategy. In a guided search, testing is performed in several rounds, with the results from earlier tests are examined and used to select a new set of parameters that are more likely to perform well. There are two main ways to guide this selection process. One approach uses prior knowledge about how the quality of a test can be measured. In other words, it relies on understanding the rule or calculation that tells us how good or bad a particular parameter set is and uses this information to choose the next points to test. The other approach is driven by data. Here, past results are analysed using machine learning techniques to predict which parameter sets are most promising for the next round of testing. Generally, guided search aims to minimize or maximize an underlying function, called the objective function. This objective function describes the problem to be solved. In this case, it is the performance of the CCAM vehicle in the logical scenario. The objective function therefore maps a combination of parameters, a concrete scenario, to a performance value based on a metric.

Guided search approaches, especially those based on data, allow for the identification of Parameter Subspaces. As such these approaches have been the major focus of development within the SUNRISE project and are discussed in Chapter 4.

3.3.1 Optimization based on logic

Methods that follow under this approach use information that is available on the problem to find improved samples. The most commonly found methods in this category are minimizers that utilize use the derivative of the objective function. An example of such a method is gradient decent. Gradient descents find the minimum by iteratively moving towards the direction that minimizes the objective functions the most from a starting point, which means in the opposite direction of the gradient of the function at the respective points. The problem with such methods is that the gradient must be known. This means that they do not work for black-box problems that are more commonly encountered in the application for safety assurance. An additional problem is the strong dependence on the starting point choice which might lead to local minima but not to the targeted global minimum.

3.3.2 Optimization based on data

Methods that follow this approach leverage machine learning approaches to learn from data collected by iteratively exploring the parameter space of the logical scenario. Common methods include the creation of a surrogate model by using a Gaussian Process Regressor or Classifier. The surrogate model is a representation of the objective function, in our case the evaluation metric, based on data collected by previous evaluations. This surrogate model can then be used by optimization approaches like Bayesian Optimization to find new sample points, which can then be used to further improve the surrogate model.

Iterative methods focus on iterative optimization techniques to obtain increasingly complex scenarios, often using accident databases as a starting point or defining criticality metrics. For instance, [27] calculates a safety zone beside the vehicle as a measure of criticality and uses evolutionary algorithms to maximize it. In [28], sensor phenomenological models are built to identify complex test scenarios for perception capacities, while other approaches include the autonomous vehicle within a feedback loop, using evaluation results to determine the next scenario. Reinforcement learning techniques for optimizing scenario criticality are also proposed in [29] and expanded upon in [30] and [31].

One approach that falls under this category is importance sampling. In importance sampling, the scenario parameters are sampled from an importance density distribution. This enables focusing the assessment of a system on scenarios in which the system-under-test is challenged. By weighting the results based on the importance density and the original distribution, one can obtain unbiased results despite sampling from a different distribution. It is, however, not straightforward to determine the importance density.

In [32] and [14], an iterative approach to determine the importance density is proposed. First, a multivariate distribution of the scenario parameters is estimated based on the parameter values of a set of scenarios that are observed in the real world. Next, simulations are conducted of scenarios in which the parameter values are sampled from the estimated probability density function. This is also known as crude Monte Carlo sampling [14]. Then, the results of the simulations are ordered according to a chosen metric that quantifies the severity of the outcome of the simulation run. This could be the minimal time-to-collision, but other so-called surrogate safety metrics are also possible. Then, based on a chosen percentile of "most important" scenarios (according to the chosen metric), earlier generated scenarios are selected. Using the same approach to determine the original distribution based on the parameter values of the real-world scenarios, the importance density is constructed using the selected scenarios instead of the real-world scenarios.

The advantage of this approach is that the importance density is based on the outcome of an initial round of simulations, thereby adapting to the capability of the system under test. In theory, this approach can be applied iteratively in order to focus more and more on the more challenging scenarios. The required initial round of simulations can also be considered as a disadvantage: this may require significant computational time. Therefore, if there is already knowledge upfront on what might be safety relevant parameter values, it may be worthwhile to do the initial round of simulations with a density based on this knowledge.

Recent Artificial Intelligence (AI) methods leverage generative or “trustworthy AI” models to create specific scenarios, often more critical or significantly different from those in real-world data. In [33], the authors provide a synthesis covering this topic, mentioning numerous works that use adversarial attacks to challenge AI models, generating hard-to-classify scenarios from real data that represent challenges for autonomous driving. In [34], an autoencoder’s reconstruction error serves as a novelty indicator for finding high-novelty scenarios, an idea expanded in [35], where the authors use Generative Adversarial Networks (GANs) and a Variational Autoencoder (VAE) to both extract existing scenarios and generate new ones from them.

3.4 Other Approaches

In this section, a sampling and selection approach is discussed that does not fit in the concretization categorisation shown in figure 5.

3.4.1 Knowledge-Based Approaches

These approaches use abstract information, such as standards, guidelines, or expert knowledge, for scenario generation, often through ontologies that represent knowledge, properties, and relationships, enabling automatic generation of test scenarios as valid combinations. For example, in [36], the authors combine a highway ontology, a weather ontology, and a vehicle ontology, modelling relationships between them. Other multi-step ontology applications are presented in [37] and [38].

4 SUBSPACE CREATION APPROACHES

This chapter gives an overview over of sampling and selection approaches that have been developed in more detail within the SUNRISE project, to be used as part of the Concretization component of the Query & Concretize block of the SUNRISE SAF.

While the last chapter has introduced general categories and selection and sampling approaches, the focus in this chapter is to present approaches that have been developed for the SUNRISE project, extended for the project, or used to demonstrate project use cases. A focus of this chapter is the use of Parameter Subspace Creation to guide the concretization approaches. This chapter starts with a section on general process steps for scenario concretization and how they can be implemented. Following section introduce different approaches, starting with an implementation of the popular LHS method introduced in the last chapter and then introducing more complex methods.

4.1 Subspace Creation Methodology

Not all the parameter space of a logical scenario needs to be tested with the same amount of effort. Some regions of the space might not be challenging, with large changes in parameters do not influence the outcome and some parts of the parameter space might contain concrete scenarios, where crashes are unavoidable. To focus testing where it is needed, a special focus is to be put on parameter Subspace Creation and a methodology to achieve this. The goal is to identify parts of the parameter space where testing is most required. It is important to mention that the subspaces do not have to be discrete and separate spaces. Another way to view them is a statistical distribution over the entire parameter space.

While our goal is the usage of methods that allow for Parameter Subspace Creation, depending on the problem simpler methods might be sufficient. However, showing that testing in a specific part of the parameter space is sufficient requires guided methods that can iteratively explore the parameter space. In this chapter, we introduce approaches to solve this problem with surrogate modelling. These approaches replicate the performance of the CCAM system with a model. The uncertainty in this model guides both the sampling in certain sub-spaces and can be used as a coverage criterion to evaluate when enough testing occurred.

4.2 Process Steps for Scenario Concretization

To instantiate logical scenarios into concrete ones, it is necessary to assign specific, feasible values to all free parameters. The presented approach considers parameters such as the initial conditions of dynamic systems and sensor configurations, which remain invariant throughout the simulation, in contrast to parameter trajectories that evolve over simulation time. The concrete instantiations of one logical scenario with N parameters can be interpreted as points in an N-dimensional cartesian coordinate frame. Due to the above-mentioned constraints, only a subset (potentially non-convex) of the N-dim. space is feasible.

Another important aspect of scenario-based testing is the definition of appropriate evaluation criteria for the scenario results. In this approach the test cases are assessed as passed or failed depending on whether one of the following metrics lies below predefined thresholds:

1. Minimum Time-to-collision (TTC): For every simulation time step the TTC between ego and other vehicles is computed. The minimum TTC value encountered during the entire simulation is used for evaluation.

2. Minimum distance: The minimum distance between ego and other vehicles that occurred during the simulation must not fall below a certain value.

For the majority of practically relevant scenarios, performing an exhaustive search within this subset is computationally infeasible, particularly when the parameters are real-valued. One possibility to solve this issue is applying techniques from statistical sampling or pseudo-random sequences with decent space-filling properties to generate a representative collection of concrete scenarios. Latin Hyper-Cube Sampling and Sobol Sequence are supported by the presented approach. Because all concrete scenarios are created independently from each other, these methods are called "open-loop" for the remainder of this section. One drawback of open-loop techniques is that they do not favour critical / failed test cases, i.e., it is a matter of computation effort (number of test cases, density of points in parameter space) and "luck" whether they are found or not.

To address this issue, closed-loop methods employ optimization techniques to identify critical parameter combinations in the feasible parameter space. Beginning with an initial estimate - comprising one or more distinct samples - the simulation outputs are evaluated using an objective function that quantitatively represents a safety metric. Based on that the next selected parameter combinations are guided towards failed test cases w.r.t. above defined criteria. Since the evaluation process involves executing the full simulation framework (including the ADF, vehicle dynamics, and related components), analytical gradients or Hessians are not available. Consequently, derivative-free optimization methods are utilized. Figure 7 depicts the principle of the approach schematically. The circular information flow within the "optimization problem solving" block motivates the term "closed-loop". The "execute" block contains the boundary for controlling the simulation environment and retrieving the necessary data for computing the safety-related objective function. Since this is a step, which typically consumes considerable amounts of time, it should be invoked as rarely as possible. To provoke failed test cases the objective function is chosen as a weighted combination of minimum-TTC and minimum-distance.

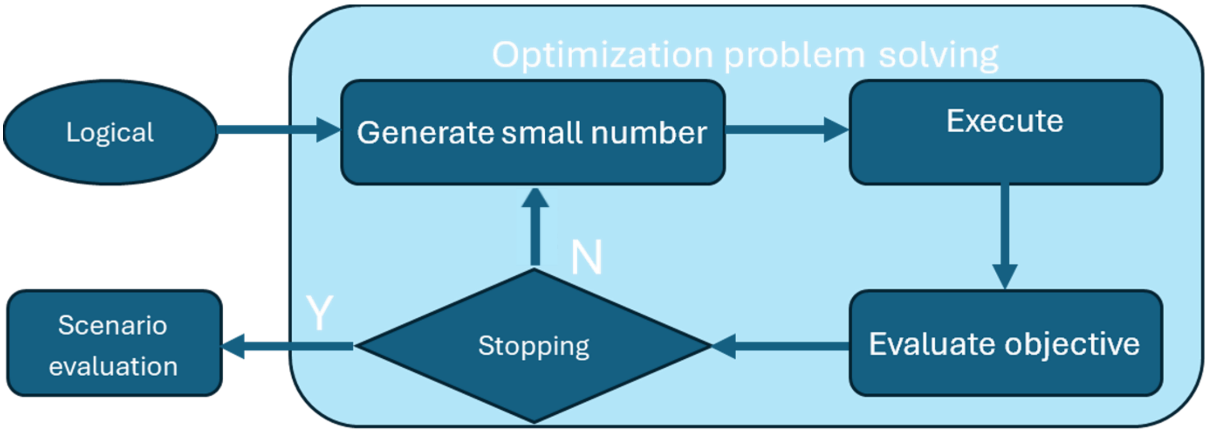


Figure 7: High-level process steps for scenario concretisation.

In the presented implementation, the user can choose from several state-of-the-art black-box optimization algorithms. Particle swarm and surrogate optimisation proved to be particularly useful regarding the number of iterations (simulation) required to reach critical cases. These algorithms are described in more detail in [39], which has been accepted for publication.

Figure 8 illustrates the results obtained from via the open-loop method for a logical scenario where the ego vehicle does a right turn and must give way to another vehicle coming from the left. Three parameters (other's speed, sensor horizontal field-of-view, and sensor range) are varied. Every marker indicates one tested concrete scenario, and the colour represents the evaluation result. The closed-loop result for the same scenario is shown in Figure 9. Due to the guided searching approach, the number of simulations is significantly smaller than in the open-loop case. It is important to mention that different scenarios and parameter space dimensions can effect the necessary amount and distribution of initial guesses. However, this was not explicitly consider in this approach.

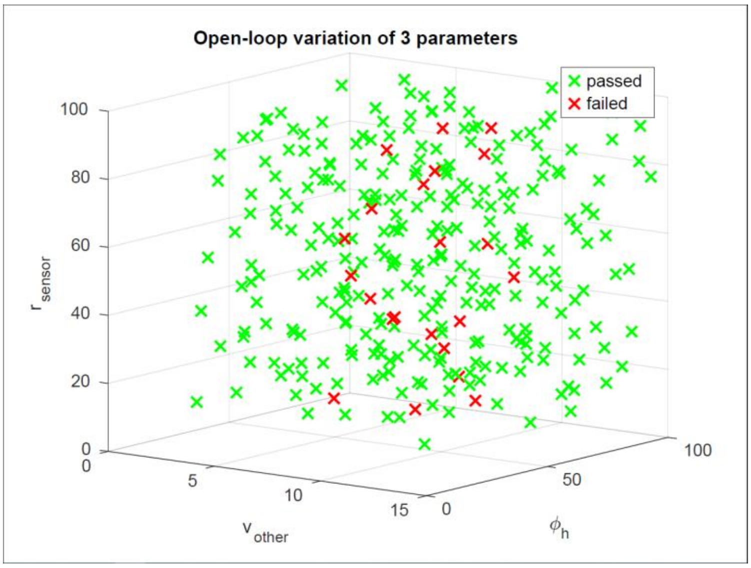


Figure 8: Open-loop test evaluation results for an intersection scenario. Each point represents one executed simulation, and the colour indicates the test case result.

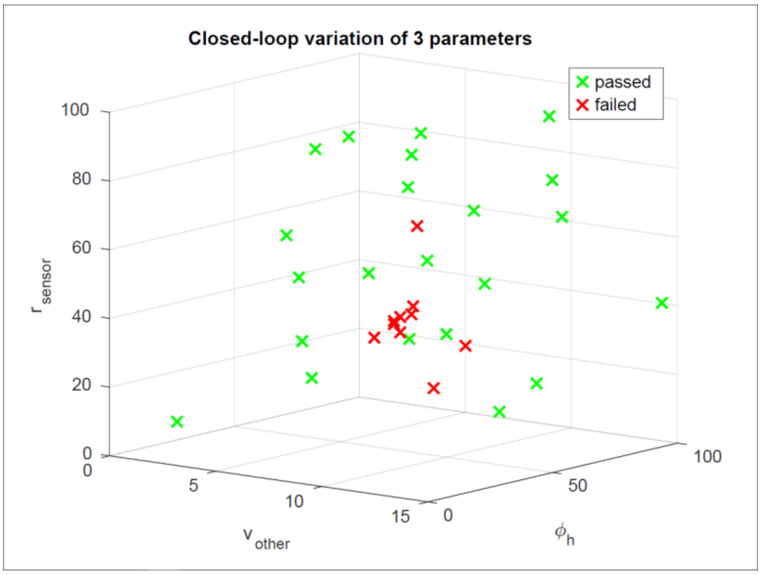
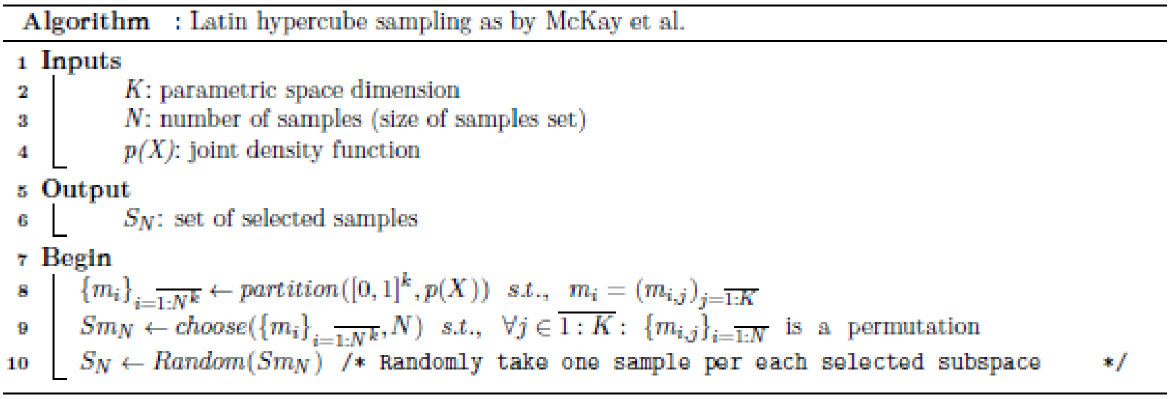


Figure 9: Optimisation-based scenario concretisation result. The approach searches explicitly for low TTC and distance values.

avoidance. The results, along with the corresponding input parameters, are then documented and passed to the Bayesian optimization module.

4.3 Constrained Latin Hypercube Sampling

This work uses a constrained Latin hypercube for sampling the simulation parameters. The main difference is that some predefined constraints are verified while generating the hypercube samples to decrease the amount of spurious generation. This choice falls in the near-random, non-distribution naïve sampling category. This choice is based on the attempt to fill the sampling space with a reduced number of samples, thereby decreasing the required number of simulations and reducing the computational cost of the analysis. We illustrate below the Latin Hypercube Sampling algorithm:



```
Algorithm : Latin hypercube sampling as by McKay et al.
Inputs
    $K$: parametric space dimension
    $N$: number of samples (size of samples set)
    $p(X)$: joint density function
Output
    $S_N$: set of selected samples
Begin
    $\{m_i\}_{i=\overline{1:N^k}} \leftarrow partition([0,1]^k, p(X))$  $s.t.,$  $m_i = (m_{i,j})_{j=\overline{1:K}}$
    $Sm_N \leftarrow choose(\{m_i\}_{i=\overline{1:N^k}}, N)$  $s.t.,$  $\forall j \in \overline{1:K}:$  $\{m_{i,j}\}_{i=\overline{1:N}}$  is a permutation
    $S_N \leftarrow Random(Sm_N)$  /* Randomly take one sample per each selected subspace */
```

A Latin Hypercube Sampling is a process that involves generating a hypercube, where each dimension represents a parameter to be sampled (e.g., the x and y starting positions of the vehicles to be simulated, their speed, their model, and the weather). Each variable is assigned a certain range of possible values (e.g., the initial speeds of the vehicles shall go from 10 to 60 km/h). Once the hypercube is created, the sampling process begins by dividing each dimension into a finite number of intervals (e.g., the possible speeds will be divided into intervals of 5 km/h each). Each of these intervals will be sampled randomly, but only once. The next step (and the last one for a classic Latin hypercube sampling) is to group the obtained samples of each variable by grouping one of the sampled initial speeds with one of the initial X positions with one of the vehicle models, etc. This way, the whole sampling space, although not entirely covered, could be significantly represented in the sampling results while reducing the number of required samples in comparison to naïve random sampling. In Figure 10, a 2D representation of a Latin hypercube is shown (a), in which each variable is divided into intervals, and a single sample is taken from each interval of each of the two dimensions (b):

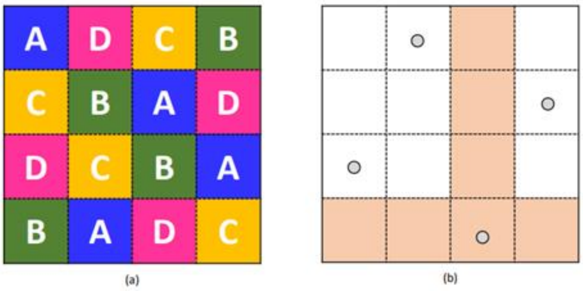


Figure 10: Latin hypercube structure.

The final step is to add some constraints to the results. Since the simulations are supposed to represent a 3D world with realistic objects in it, there are some limits to what is worth simulating. In particular, the initial positions of the vehicles as well as their models are considered and checked to see if any of the generated samples would cause the overlapping of two or more vehicles. This situation is not feasible in the real world, so it would not make sense to simulate it. Therefore, these sampling pairs are rejected, and those samples are reassigned to another group. If it is impossible to find a valid group for those samples (which rarely happens), the space is resampled for new variable values.

In Figure 11, several samplings are shown. The dots represent the starting position of the simulated vehicles (the horizontal axis represents the width of the road starting from a fixed coordinate (here represented as 0). In contrast, the vertical axis represents the road's longitudinal dimension). The rectangles are a representation of the size of each vehicle. If the dots are blue, the sampling is feasible and therefore valid. If the dots are red, an overlap in some of the vehicles was detected, so the sampling is considered invalid, and either a re-grouping or re-sampling will be performed.

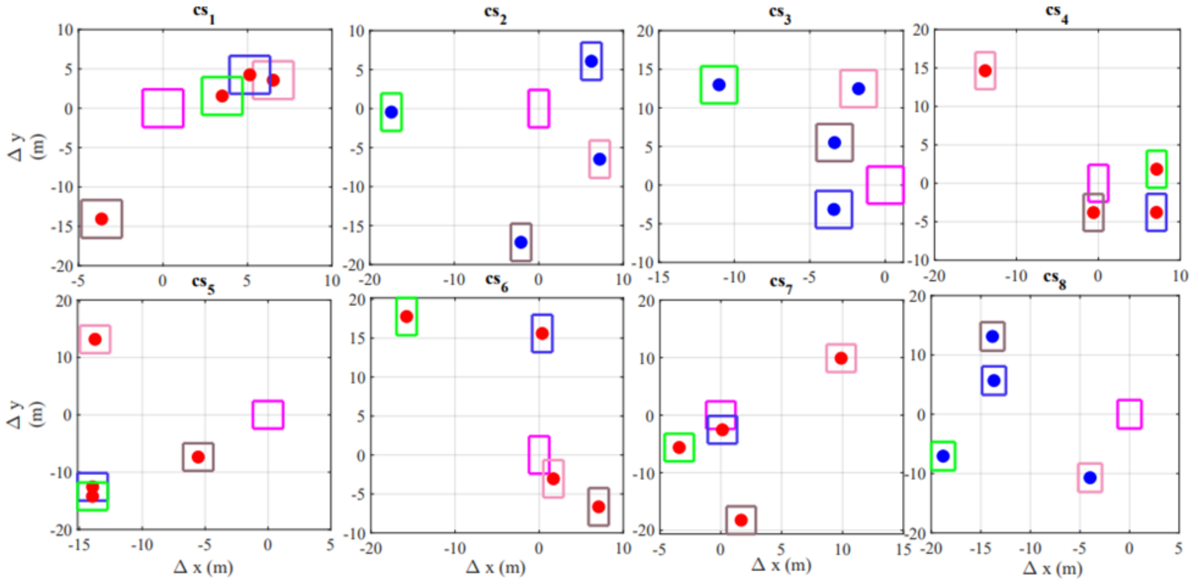


Figure 11: Latin hypercube samplings.

Figure 12 shows the result of the constrained Latin hypercube sampling. In it, all the sampled groups are valid because there is no longer any overlap between vehicles in any of the configurations. Therefore, all the points are shown as blue:

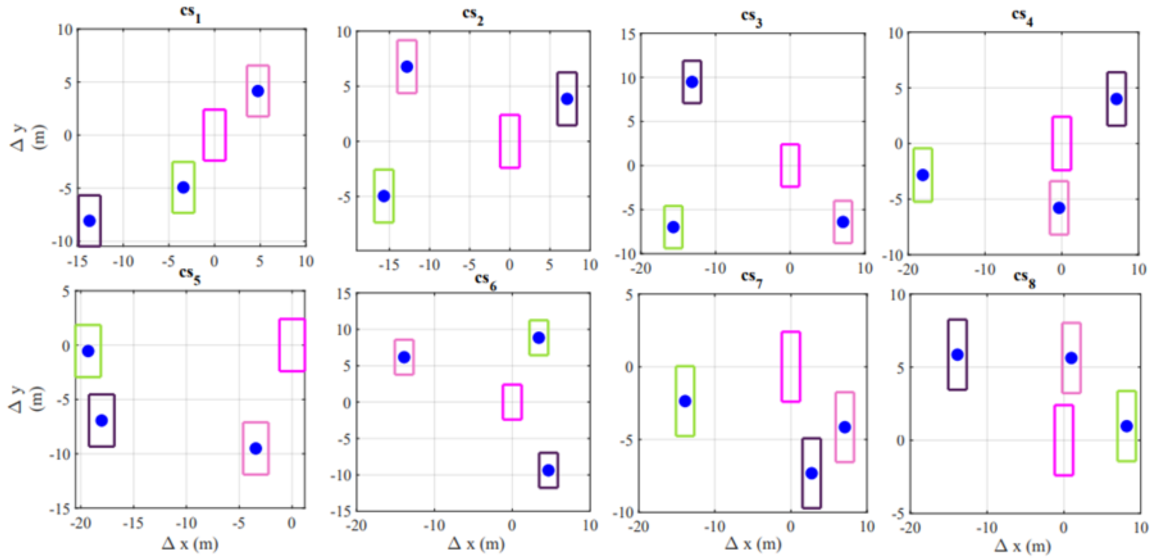


Figure 12: Constrained Latin hypercube sampling results.

4.4 Method for boundary point detection

As mentioned in the paragraph on importance sampling in section 3.3.2, one of the challenges in scenario-based testing is selecting interesting points from the parameter space of a logical scenario when the parameter space is too large for exhaustive search. Typically, a specific pass-fail criterion is linked to the system under test and the logical scenario. This criterion evaluates the outcome of a concrete scenario and determines whether the system's behaviour is safe or unsafe. The goal of this method is to focus on boundary cases in the parameter space, where minor variations in a parameter vector can determine pass or fail outcomes.

These points lie at the boundary between the pass and fail regions and likely expose the system under test to a situation it can “barely” or “nearly” handle. Points away from the boundary are likely less interesting because the system either passes without facing a challenge or fails without having a chance.

The following method has been first introduced in two master theses, [40] and [41]. The most important prerequisite to apply this methodology is that a first sampling was already applied, or first data is available already covering the parameter space sufficiently. Sufficient in the sense of having no big gaps in the parameter space and boundary points exist that can be determined by this method. Additionally, the result data must be divided into two results e.g., safe and unsafe. This method relies on a cost function returning binary output and the method is therefore independent of the metrics used, the thresholds of the metrics, and how they were combined within the cost function e.g., AND, OR, and XOR condition. can be applied iteratively several times. The main goal of this methodology is to increase the sampling density surrounding the boundary points and not to detect gaps in the sampling region. Therefore, if already the first data does not cover a region of unsafe sample points, also an iterative application of this methodology will also more likely not identify these. But this depends on the sampling methodology applied for the resampling, which could also include other points in the parameter space, except for the boundary conditions returned by this methodology. The prerequisite for the sampling methodology for the resampling is the inclusion of boundary conditions.

To describe this procedure in more details, assume a sample of parameter vectors is drawn from the parameter space of the logical scenario, and each parameter vector is simulated and evaluated by the pass-fail criterion. Figure 13 depicts two examples where the safe sample points are shown in green, the unsafe ones in red, and the boundary between safe and unsafe regions in blue.

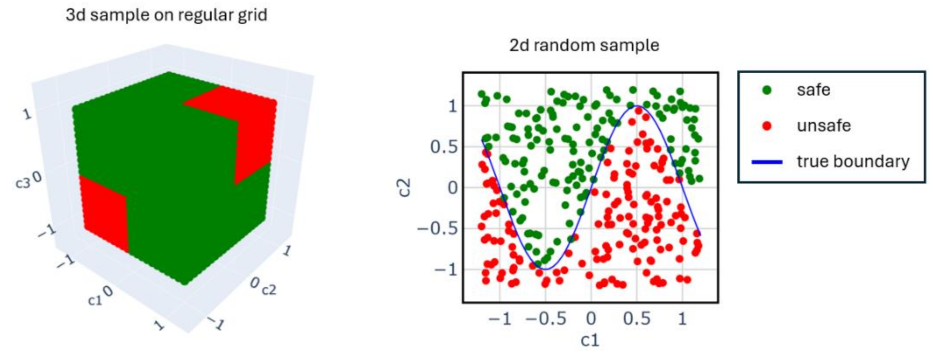


Figure 13: Examples of samples with safe points in green and unsafe ones in red.

The goal is to detect the boundary (or uncertain) points at the boundary between safe (green) and unsafe (red) regions. Here, a safe point is a boundary point if one of its “neighbours” is unsafe, as shown in Figure 14. The concept of “neighbours” is described in the following paragraph by looking at the intuitively clear case of regular grids and introducing a small adjustment which is applicable to randomly chosen samples.

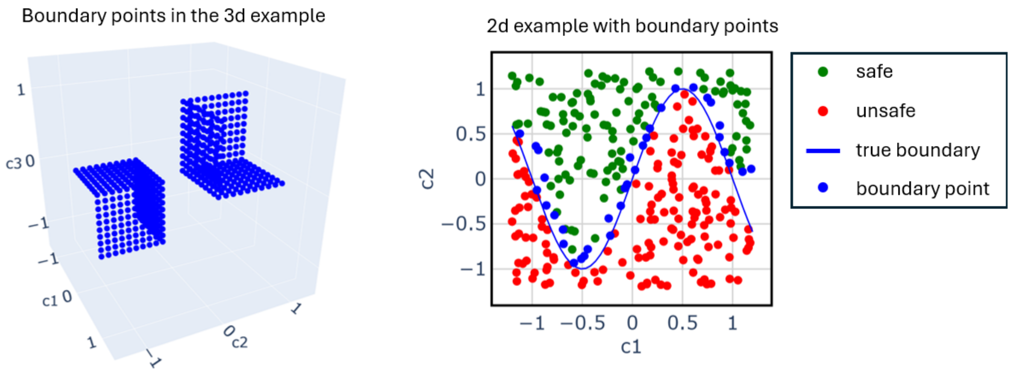


Figure 14: Samples with boundary points in blue.

If the sample points are located on a regular grid as in the left example of Figure 13, it is intuitively clear that two distinct points are neighbours if they differ by at most the step length in each coordinate. In this case, one can check the $3^d - 1$ d Figure 15 for two and three dimensions, the selected point (green) is a boundary point if it is safe and any of its neighbours (black) is unsafe. In the figure, h1, h2, and h3 denote the step lengths of the axes.

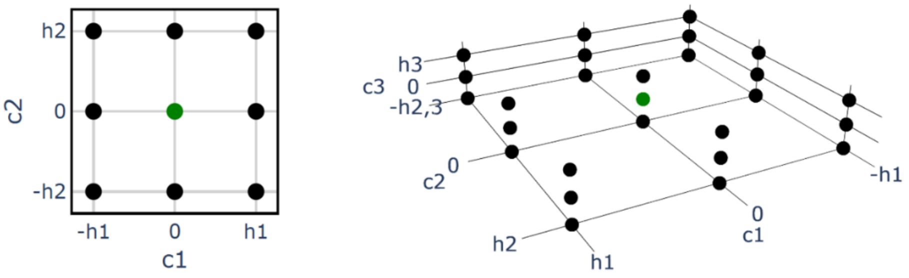


Figure 15: Neighbour points (black) of the origin (green) in regular grids in 2D (left) and 3D (right).

This idea was used in the left example of Figure 14. However, $3^d - 1$ grows exponentially with the dimension; thus, one might only check the $2 \cdot d$ neighbours along the parameter axes in high-dimensional spaces. In randomly chosen sampling, such as in the right example of Figure 13, it is not inherently clear which points are the neighbours of a given point. The idea here is to search neighbours along (or close to) the coordinate axes. To this end, assume in the following that the parameters are adimensional or normalized (so that parameters of potentially different physical units can be compared) and a metric for measuring the distance between sample points (such as the Euclidean norm) is specified. Then, for example, along the negative first axis, $\vec{q}$ is defined to be a neighbour of $\vec{p}$ if the difference $\vec{d} = \vec{q} - \vec{p}$ satisfies $-d_1 > \max \{|d_j|;\ j = 2,\ \ldots, d\}$ and $\vec{q}$ is the closest point to $\vec{p}$ that satisfies this condition. The purpose of the condition on d_1 is to ensure that “main direction” of $\vec{d}$ is the negative first axis in the sense that d_1 is negative and its absolute value is bigger that of other coordinates. To see that it makes sense, Figure 16 depicts an example with four points in the xy-plane. The region of points satisfying the condition $-d_1 > |d_2|$ is shown in grey (the “main direction” of $\vec{d}$ is the negative x-direction). Thus, $\vec{q}$ is the neighbour of $\vec{p}$ in negative x-direction since $\vec{r}$ is farer away from $\vec{p}$ and $\vec{s}$ lies in the search region for neighbours in positive y-direction.

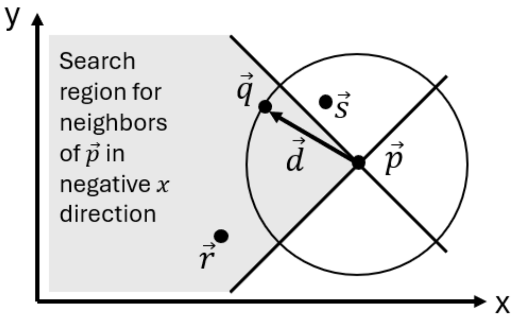

y
Search region for neighbors of $\vec{p}$ in negative x direction
$\vec{q}$
$\vec{s}$
$\vec{d}$
$\vec{p}$
$\vec{r}$
x

Figure 16: 2d example for searching neighbours of a point $\vec{p}$ in negative x-direction.

This idea is repeated along all directions and for each point in the current sample. As a result, the boundary points in the current sample are identified such as the blue points in Figure 14. As mentioned at the beginning of this section, the identification of these points is the goal of this method. Thus, the investigation of the current sample is finished.

Optionally, the boundary identification method can be applied iteratively in order to increase the precision of the boundary. However, such an iterative process relies on a re-sampling strategy. Such a re-sampling strategy should aim at making the currently identified boundary more precise and at revealing unseen boundaries. $h > 0$ for the safe-unsafe-boundary and $H > 0$ for whole parameter space of the logical scenario. Then, an example of a re-sampling strategy consists of the following three steps:

1. As a first step, to make the currently identified safe-unsafe-boundary more precise, proceed as follows:
    - For each current boundary point and each direction, check whether the distance to the neighbour is larger than h .
    - If yes, generate a new sample point inside the search region at a distance smaller than h .
    - If the current point has no neighbour in the current direction and if the distance to the outer boundary of the parameter space is larger than h , then also generate a new sample point inside the search region at a distance smaller than h .
2. As a second step, to (hopefully) reveal new safe-unsafe-boundaries, replace the boundary points by the non-boundary points (both, safe and unsafe ones), replace h by H , and repeat the first step. Optionally/Alternatively sample (e.g. uniformly) from the whole parameter space.
3. As third step, evaluate each new sample point as safe or unsafe by the pass-fail-criterion, add these points to the current sample, and start the next iteration.

As final remark, notice that this method is limited to continuous data and not applicable for discrete (unsorted/nominal) data (e.g., what is the neighbour of sunny weather conditions?).

In summary this method has the following limitations:

- This method describes one sampling iteration step rather than a complete sampling method, which means another method for initial sampling and optionally for potential resampling is required.

- Method not applicable for unsorted discrete data (neighbours and boundary unclear)

- Method relies on safe-unsafe characterisation of data points (e.g., by the usage of a cost-function with all its limitations)

- Data needs to be normalised, and neighbour points are found by applying distance metrics (e.g., equivalent distances to angular neighbour points and velocity neighbour points)

- Description (especially machine-readable) of multidimensional boundary volume (multidimensional boundary surface of specific thickness) for potential resampling not defined (outputs of this method are clusters described by the boundary sample points identified)

4.5 Equiprobable partitioning of the parameter space

One of the key challenges in scenario-based CCAM testing is selecting a meaningful and representative subset of scenarios from a vast parameter space, where exhaustive sampling is infeasible. This approach addresses this challenge by leveraging deep generative models to construct structured subspaces, enabling efficient scenario selection and optimal coverage assessment. This approach is rooted in transforming the high-dimensional representation of driving scenarios into a latent space with a known probability distribution, facilitating a rigorous evaluation of dataset completeness. Unlike traditional scenario selection methods that rely on fixed parameter grids or probabilistic sampling techniques such as Monte Carlo or Latin Hypercube Sampling, the approach introduces a novel paradigm that ensures a structured, equiprobable partitioning of the scenario space, significantly improving selection efficiency and test relevance.

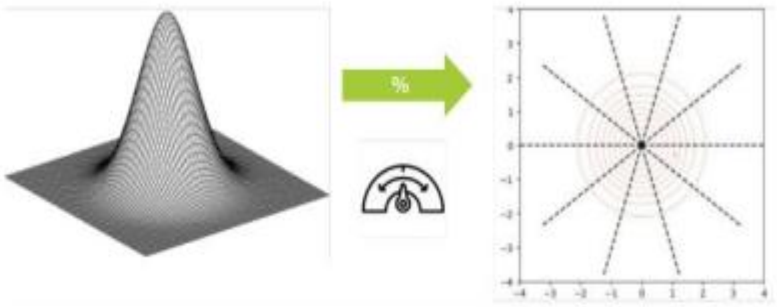


Figure 17: Utilizing a 2D Gaussian distribution to create equiprobable zones with equidistant circles and the radii method for achieving representative sampling.

As depicted in Figure 17 the transformation of scenarios into a latent space enables a structured partitioning, where the known Gaussian distribution allows for equiprobable sampling of test cases. The method relies on Variational Autoencoders (VAEs) to convert the unknown distribution of real-world driving scenarios into a structured latent space, ensuring statistical consistency in scenario selection. A VAE is a type of neural network architecture used for unsupervised learning, that learns a probability distribution over a latent space. The transformation performed by the VAE allows for the definition of subspaces where critical and non-critical points are clearly delineated, enhancing the ability to detect underrepresented driving situations.

To further refine the completeness assessment, the approach employs an equiprobable discretization strategy, dividing the latent space into zones of equal probability mass rather than using arbitrary parameter grids. This approach, illustrated in Figure 18, ensures that scenario sampling maintains statistical representativeness while reducing redundancy.

The first methodological step transforms the unknown, high-dimensional distribution of real-world driving scenarios into a known, structured latent space. This is achieved using a Factor β-Variational Autoencoder (β-VAE), which balances reconstruction accuracy with disentanglement and normality of latent features. A cyclical KL-annealing schedule is used to stabilize training, enabling the model to produce a centred, reduced 2D Gaussian distribution in the latent space. This choice of latent representation ensures feature independence and radial symmetry, which are crucial for the next step.

Once in this 2D Gaussian latent space, the continuous representation is discretized into equiprobable zones, areas of equal probability mass, rather than arbitrary parameter grids. This is implemented by dividing the latent space into concentric circles of equal probability, intersected by evenly spaced radii, directly yielding a set of equiprobable partitions. This structured partitioning guarantees statistical representativeness and avoids redundancy in the test set.

The required number of zones is determined statistically using the Central Limit Theorem (CLT). For a target confidence level and margin of error, the necessary sample size n is calculated from:

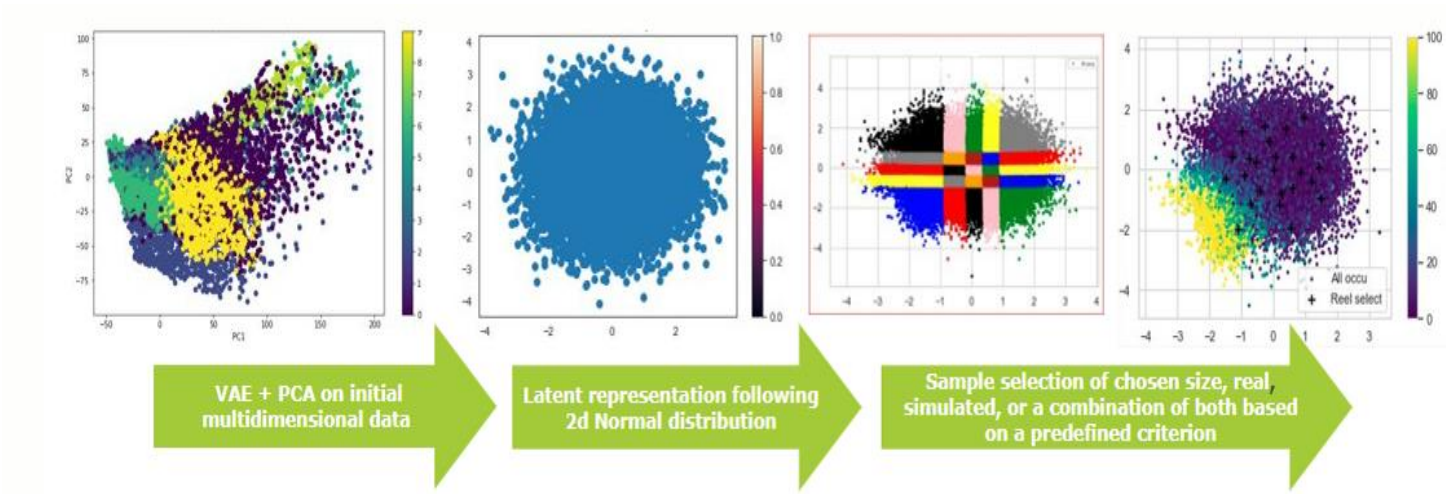
$$n = \left(Z_{\alpha/2} \frac{\sqrt{p(1-p)}}{E} \right)^2$$

where p is the estimated occurrence rate, E the desired precision, and $Z_{\alpha/2}$ the z-score for the confidence level. For example, with$p = 0.002$, $\mathrm{E} = 0.001$, and $\mathrm{Z} = 3.29$ (99.9% confidence), the required sample size is approximately 21,605. The number of equiprobable zones is then matched to this target.

The final step assesses coverage by counting non-empty zones in the latent space. Empty zones are decoded back to the original parameter space via the VAE decoder, identifying missing scenarios and enabling the generation of synthetic test cases. This is particularly valuable for capturing rare but safety-critical events such as near-miss collisions or sudden cut-ins.

Missing scenarios can be identified through an inverse mapping of the latent space back to the original scenario parameterization, enabling the generation of synthetic test cases to fill the gaps in underrepresented regions. This process is particularly valuable in ensuring that rare but safety-critical driving situations, such as near-miss collisions or sudden cut-ins, are adequately represented in validation datasets.

The steps of this methodology are described in more detail in [42].



VAE + PCA on initial multidimensional data

Latent representation following 2d Normal distribution

Sample selection of chosen size, real, simulated, or a combination of both based on a predefined criterion

This solution is a Vedecom-MOOVE4.0 project outcome brought as a background to Sunrise.
VEDECOM's Intellectual Property.

Figure 18: Steps in the methodology.

By integrating deep generative modelling with a robust statistical sampling methodology, the presented approach significantly enhances the effectiveness of scenario-based CCAM testing, as shown in [42]. It not only improves the efficiency of scenario selection but also ensures that the test set captures a diverse and representative range of driving situations, ultimately strengthening the safety validation process for autonomous driving systems.

4.6 Active design of experiment method

Active Design of Experiments (DoE) Sampling Method is an innovative approach to scenario sampling and selection. This methodology, categorized as an adaptive and guided search technique, aims to achieve maximum coverage of scenarios while minimizing the number of simulation-runs. It places particular emphasis on critical and safety-relevant areas within the scenario space.

The methodological concept behind Active DoE distinguishes it from classical random sampling or space-filling techniques, such as Combinatorial Sampling. Instead, Active DoE employs a closed-loop learning process that integrates machine learning surrogate models with scenario sampling. This integration allows for continuous evaluation of the scenario space, identifying regions that are underrepresented, uncertain, or crucial for understanding system behaviour.

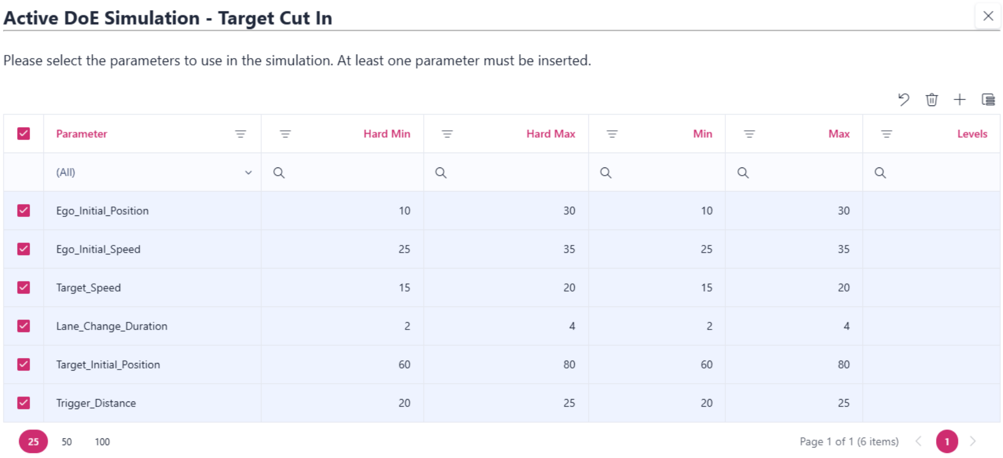
The overarching goal of Active DoE is to prioritize testing in areas that have the most significant impact on key performance indicators, such as safety and comfort. By focusing resources on high-impact regions, this method effectively reduces unnecessary simulation efforts in low-risk areas.

Active DoE is used within the SUNRISE project with the AVL CAMEO™ software, and this approach is seamlessly integrated with AVL Scenius SCDB, facilitating easy configuration and triggering of the concretization process within a SCDB. This integration enhances the efficiency and effectiveness of scenario testing in various applications.

4.6.1 Step-by-Step Workflow of Active DoE Sampling:

Definition of Scenario Parameter Space:

The first step in the Active DoE process involves defining the overall scenario parameter space. This means identifying all the input parameters that describe the variability of the driving scenarios. These parameters may include Ego and Target vehicle initial speeds, Lateral and longitudinal positions and environmental conditions (e.g., road friction, weather)



Active DoE Simulation - Target Cut In

Please select the parameters to use in the simulation. At least one parameter must be inserted.

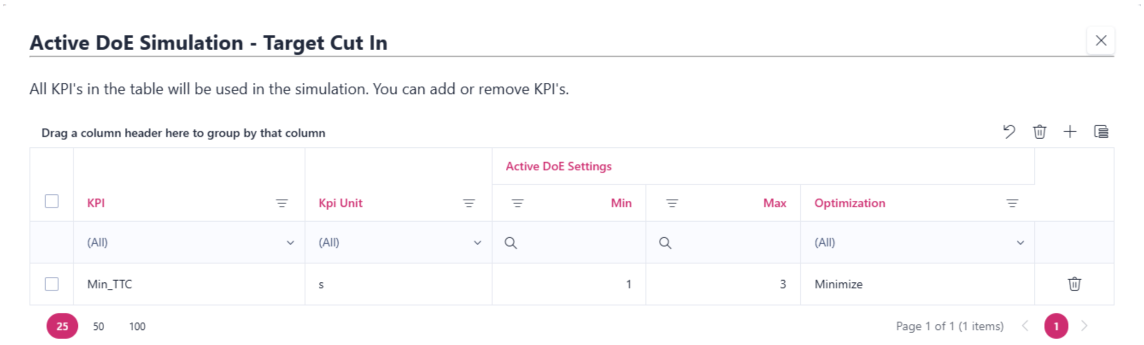
☑	Parameter	Hard Min	Hard Max	Min	Max	Levels
	(All)					
☑	Ego_Initial_Position	10	30	10	30	
☑	Ego_Initial_Speed	25	35	25	35	
☑	Target_Speed	15	20	15	20	
☑	Lane_Change_Duration	2	4	2	4	
☑	Target_Initial_Position	60	80	60	80	
☑	Trigger_Distance	20	25	20	25	

Figure 19: Parameter Space Definition in Active DoE.

Definition of the Critical Region

Once the full scenario space is defined, the next step is to identify which regions of this space are considered critical. Critical regions are defined as areas where the ADAS/AD system performance is sensitive, risky, or likely to produce safety-relevant outcomes (e.g., near-miss situations).

The remaining scenario space, where the system is known to behave safely and predictably, is marked as the non-critical region. Time-to-Collision can be used in order to assess the criticality. On the example below the critical region for the Cut-In scenario has been chosen where the minimum TTC is between 1 and 3 seconds, and the optimization will try to minimize the TTC to 1 second.



Active DoE Simulation - Target Cut In

All KPI's in the table will be used in the simulation. You can add or remove KPI's.

Drag a column header here to group by that column

□	KPI	Kpi Unit	Active DoE Settings		Optimization	
			Min	Max		
	(All)	(All)			(All)	
□	Min_TTC	s	1	3	Minimize	

25 50 100

Page 1 of 1 (1 items) 1

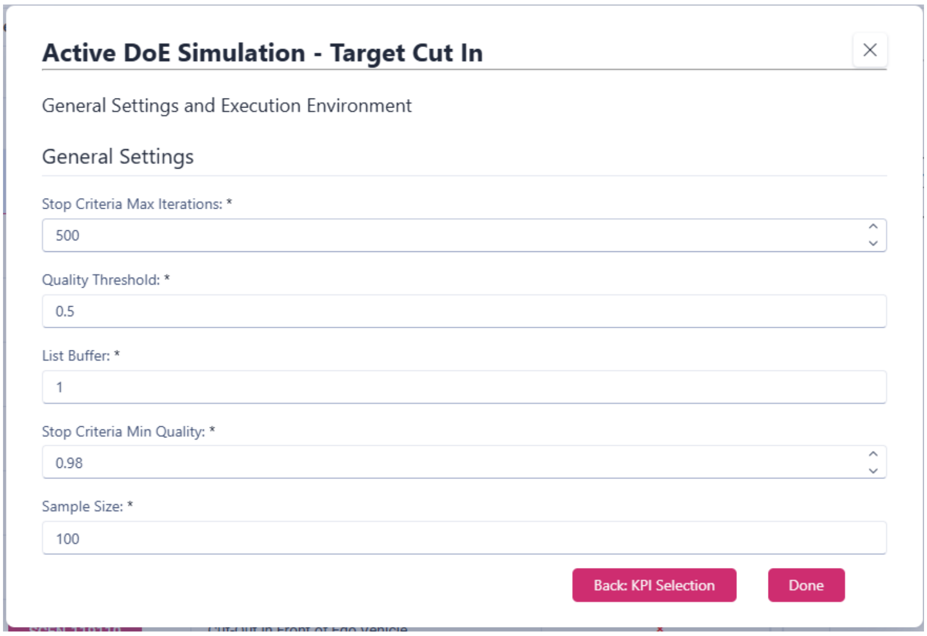
Figure 20: Critical Region Definition in Active DoE.

4.6.2 Configuring the sampling iterations

Active DoE method allows detailed configuration of the sampling, stopping criteria, and model quality settings.

Table 1: Active DOE Configuration.

Parameter	Description
Stop Criteria – Max Iterations	This setting limits the maximum number of Active DoE sampling iterations. After each iteration, the surrogate model is retrained, and new points are selected. The process stops when either this maximum iteration count is reached or earlier if quality thresholds are satisfied.
Quality Threshold	The minimum value for the model quality to be considered by the strategy. If a model has a calculated quality below this value it is considered useless.
Stop Criteria - Min Quality	This is a hard stop quality level. Even if the Max Iterations limit is not reached, the process will automatically stop when the surrogate model achieves at least this minimum quality threshold across all relevant KPIs. This prevents unnecessary simulation runs once the desired model quality is reached.
Sample Size	This defines the total number of resulting scenarios.



Active DoE Simulation - Target Cut In

General Settings and Execution Environment

General Settings

Stop Criteria Max Iterations: * 500

Quality Threshold: * 0.5

List Buffer: * 1

Stop Criteria Min Quality: * 0.98

Sample Size: * 100

Figure 21: Simulation Configuration in Active DoE.

Initial Sampling and Model Building:

Active DoE begins with an initial, sparse sampling of the scenario space. These initial simulations are used to train a machine learning-based surrogate model that predicts system behaviour across the entire space.

Model Quality Metric:

To assess the surrogate model's performance, we apply a Normalized Accuracy Score that reflects how closely the model predicts safety-relevant KPIs:

$$Q = 1 - \frac{|\,\hat{y} - y\,|}{range(y)}$$

Where:

- y: Ground truth KPI from simulation
- $\hat{y}$: Surrogate model prediction
- $range(y)$: Acceptable KPI range

This score:

- Improves as the model's prediction approaches the actual simulation result
- It is bounded between 0 (poor) and 1 (perfect)

Adaptive Resampling and Model Refinement:
The surrogate model is iteratively refined by focusing additional simulations on regions where:

- The model’s prediction error is high.
- System behaviour is highly non-linear or sensitive.

Objective Stopping Criteria:
Sampling continues until objective, KPI-based quality metrics indicate that both critical and non-critical regions are sufficiently modelled according to the Normalized Accuracy Score metric.

Final Model Validation:
Once the model quality meets predefined targets, the final selected scenario set is used for detailed system testing and validation.

Concretizing UC 2.1 Target Cut In Scenarios with Active Design of Experiments

We used the SAF UC 2.1 TJA Target Cut-In scenarios for experimenting with the AVL-Approach. The full Factorial design spans 4 000 test cases (20 × 20 × 5 grid of Ego Speed, Target Speed and Deceleration). Within this space, the safety-critical region is where Ego Vehicle Speed and Target Deceleration Rate values are higher while Target Vehicle Speed values are slower. The criticality metric used for evaluation is Minimum Time to Collision values lower than 2 seconds. The total number of safety relevant concrete test cases are 17 (0,425% of the total). By contrast, the AVL procedure, through adaptive sampling iterations, identified 15 of these critical points (covering 88 % of the critical region) and discarded the remaining non-critical cases. This targeted approach demonstrates how Active DoE can reduce simulation effort by two orders of magnitude while still achieving near-complete coverage of the most dangerous, safety-relevant subspace.

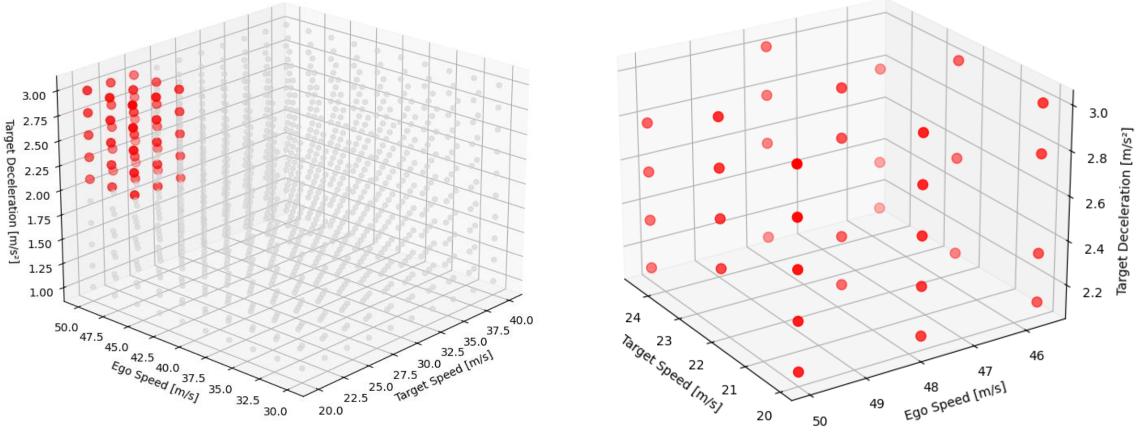


Figure 22: Full Factorial vs Active Design of Experiments Results.

4.7 Bayesian Optimization Approach

Bayesian optimization is used to optimize objective functions that are time-consuming to evaluate. It employs Gaussian Process (GP) models to derive a prior over the black box function being optimized. The GP models assume the function values to be random variables modelled by a joint Gaussian distribution. The mean and variance of the distribution are estimated for unobserved values, and an acquisition function is used to sample the surrogate function, deriving a posterior distribution.

The pass criteria, derived from System-Theoretic Process Analysis (STPA), are treated as the optimization objective. The objective function, denoted as ρ(w), represents the pass criteria in terms of exact parameter values. The optimization algorithm samples parameters to ensure $\rho(w) < 0$, indicating a violation of the pass criteria. This involves logical combinations of constraints (predicates) that are smooth and continuous functions of a trajectory ξ.

The chosen acquisition function is Expected Improvement, which balances exploration and exploitation to find a global minimum. In the context of Bayesian optimization exploration refers to sampling points in areas of high uncertainty to discover potentially better solutions, while exploitation focuses on sampling where the model already predicts high performance. The EI function balances these by favouring points that either have a high chance of improvement or are uncertain enough to be worth investigating. The utility function rewards improvements over the minimal value observed so far.

The test scenario generator aims to find parameter values that refute the pass criteria ρ. The process involves:

- Deriving Parameter Sets: The pass criteria are converted into a set of predicates, each representing a constraint. These predicates are combined using logical operations to form the objective function ρ.
- Bayesian Optimization: The algorithm iteratively samples parameter values using the acquisition function to minimize ρ. Each sample is evaluated, and the results are used to update the GP model.
- Iterative Sampling and Evaluation: The process continues until the pass criteria are violated or the time budget is exhausted.
- Identifying Multiple Minima Regions

To identify multiple minima, the algorithm eliminates already explored minima from the search space by squaring the function, making all minima rise above zero. The search continues in each parametric direction, recursively eliminating regions containing minima. This process ensures the identification of all regions containing minima.

The **first algorithm**, Bayesian Optimization for Failure Identification, aims to pinpoint the parameter values, for the identified set of parameters that lead to failure in automated vehicle testing scenarios. Inputs identified from the STPA analysis are fed into the Bayesian Optimization module, which iteratively determines parameter values such as pedestrian speed, angle, vehicle speed, and deceleration rate that could cause failure. After each iteration, the samples are sent to a simulator for stochastic evaluation. This process continues until a specification is violated or a predefined coverage threshold is reached. The primary goal is to find a global minimum, representing one of many test cases that lead to failure.

Algorithm 1: Sampling Using Bayesian Optimization

Input:
Bounded Parameter vector **W** from STPA Step 3
Pass criteria from STPA Step 2
Initial Condition vector: **I**

```
procedure DERIVEPARSETREE(PassCriteria)
    $\rho \leftarrow PassCriteria$
    for < each predicate $\mu$ in $\rho$ > do
        $BayesianOptimization(\mu, W, I)$
    end for
end procedure

procedure BAYESIANOPTIMIZATION($\mu$, W, I)
    for < n = 1,2, ... > do
        Select new $w_{n+1} \epsilon W$ by minimizing aquisition function $\alpha$
        $w_{n+1} \leftarrow \underset{w \epsilon W}{\arg min} \alpha(w)$
        $D_{n+1} \leftarrow D_n(w_{n+1}, y_{n+1})$
    end for
    return $w_i$ for which we get the minimum $y_i$
end procedure
```

Comments:
α: Expected Improvement (acquisition function)
D_n: Observed (input, output) pair from sampling
y_i: Output of the system for the i^{th} input

The **second algorithm**, Recursive Search for Multiple Minima, is designed to identify multiple minima of a continuous function, each representing different failure conditions. We propose a methodology to direct the search to find other minima of the continuous function $f: W \rightarrow R$, where W is a compact subset of R^d, and previously explored minima are eliminated by squaring the function, which raises all minima above zero. Zeros surrounding the minima in each parametric direction are identified, and the valleys containing the minima are excluded from further consideration. A new search begins recursively in each parametric direction. The search occurs over an n-dimensional hyperspace defined by parameters W1, ..., Wn, with a stack (regionsStack) storing regions to be explored, starting with the entire region. After discovering each minimum, the region is split to eliminate the valley containing the minimum, using hyper-rectangular abstractions around the minima. The algorithm terminates when regionsStack is empty. A step size (λ) is used to demarcate the region surrounding the discovered minimum, defining rectangular constraints for elimination. New search regions are constructed for each dimension based on zero points and bounds. If no zero is found within a distance λ, the search resumes with a larger λ. The choice of λ is a hyper-parameter set based on domain knowledge

Algorithm 2: Sampling Using Bayesian Optimization

Input: λ

Global Variable:

$regionsStack \leftarrow [[W_{1LB}, W_{1UB}] \ldots, [W_{nLB}, W_{nUB}]]$

```
procedure FINDMULTIPLEMINIMA ($\lambda$)
  while $true$ do
    if $regionsStack$ is empty then
      return $minimaList$
    end if
    $region \leftarrow pop(regionStack)$
    $minPoint\ BaysOptMinPoint(region)$
    for $< i = 0, 1 \ldots dimension >$ do
    $cutoffRegL\ minPoint[i] - \lambda$
    $lowB_i\ findZeroPnt(cutoffRegL, i, 0)$
    $cutoffRegL\ minPoint[i] \pm \lambda$
    $upB_i\ findZeroPnt(cutoffRegU, i, 1)$
    $zeroPnts.add([lowB_i, upB_i])$
    end for
    $omittedRegions.add(zeroPnts)$
    for $< i = 0, 1\ \ldots dimension >$ do
      $newReg1.add(zeroPnts[0..(i-1)])$
      $newReg2.add(zeroPnts[0..(i-1)])$
      $newReg1.add([regLBound, zeroPnts[i][0]])$
      $newReg2.add([zeroPnts[i][1], regU\ Bound])$
      $newReg1.add(region[(i+1)..dimension])$
      $newReg2.add(region[(i+1)..dimension])$
    end for
    if $newReg(1{,}2) not\ in\ omittedRegions$ then
      $regionStack.push(newReg1, newReg2)$
    end if
  end while
end procedure
precedure $FINDZEROPNT(cutoffRegion, i, lowUpFlag)$
  while $not\ nearest$ 0 do
    $zeroPnt \leftarrow BaysOptZeroPnt(cutoffRegion)$
    $bound_i \leftarrow zeroPnt[i]$
    if $lowUpFlag\ equals$ 0 then
      $cutoffRegion[i][0]\ bound_i$
    else
      $cutoffRegion[i][1]\ bound_i$
    end if
  end while
  return $bound_i$
end procedure
```

The methodology is applied to a case study involving dynamic elements like pedestrian speed, angle, subject vehicle speed, and deceleration rate. Bayesian Optimization identifies parameter values leading to failure conditions, demonstrating the effectiveness of the approach in finding global minima and other failure-inducing test cases.

This section discusses the results of experiments on non-convex functions and test scenarios.

A. Identifying Multiple Minima Regions for Non-Convex Functions

The first tested function is the Holdertable function, defined by

$$f(x,y) = -\left|sin(x) * cos(y) * e^{(1-\surd(x2+y2)/\pi)|}\right|$$

which causes four global minima to occur at (x,y) = (8.05502,9.66459), (8.05502,−9.66459) (−8.05502,9.66459), (−8.05502,−9.66459) for all of these f(x,y) = −19.2805.

The second function is the Eggholder function defined by

$$f(x,y) = -(y+47) * sin\sqrt{|y+x/2+47|} - x * sin\sqrt{|x-(y+47)|}$$

This function has one global minimum at f(512,404.2319)=−959.6407. Both functions have multiple local minima.

The third function is of the form,

$$f\ (x_1, x_2\ \dots x_n)\ = x_1 * x_2 * \dots .\ * x_n$$

where the number of minima increases with increasing dimension as 2d-1. The algorithm is tested up to 10th dimension.

The algorithm's performance on different non-convex functions is summarized in **Error! Reference source not found.** showing the domain, step size λ, and identified minima regions for each function.

Table 2: Performance of Algorithm 2 on different nonconvex functions.

Function	Domain	λ	Identified Minima Regions
1	x:[−10,10], y:[−10,10]	2	56
2	x:[0,512], y:[0,512]	100	275
3	Each x_i:[−50,50]	50	512

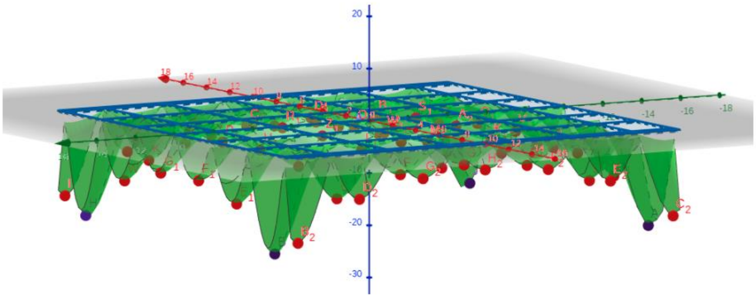


Figure 23: Visualization of the Holdertable function after running Algorithm 2, showing identified global and local minima, and eliminated regions.

B. Testing the Pedestrian Scenario Generated from STPA

Scenario Modelling: A scenario generated from STPA is modelled in IPG Carmaker. The domain of uncertainty includes:

- Subject vehicle velocity: [40 km/h, 60 km/h]
- Pedestrian speed: [5 km/h, 20 km/h]
- Pedestrian crossing angle: [0 °, 10 °]
- Deceleration rate after stop command: [3 m/s², 6 m/s²]

Objective Function: The Euclidean distance between the car and pedestrian is minimized, with a collision occurring when $\sqrt{(car.y\ - ped.y)^2 + (car.x\ - ped.x)^2}$. A safety envelope of 5 m is considered longitudinally (in the direction of the road, ahead of the ego).

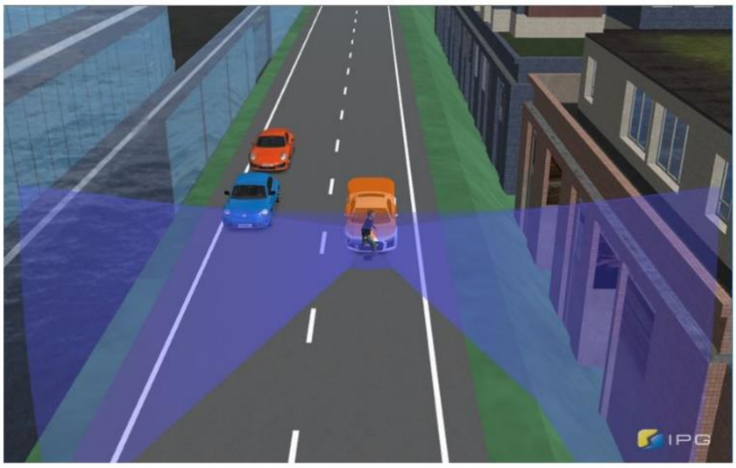


Figure 24: A visualisation of the simulated collision.

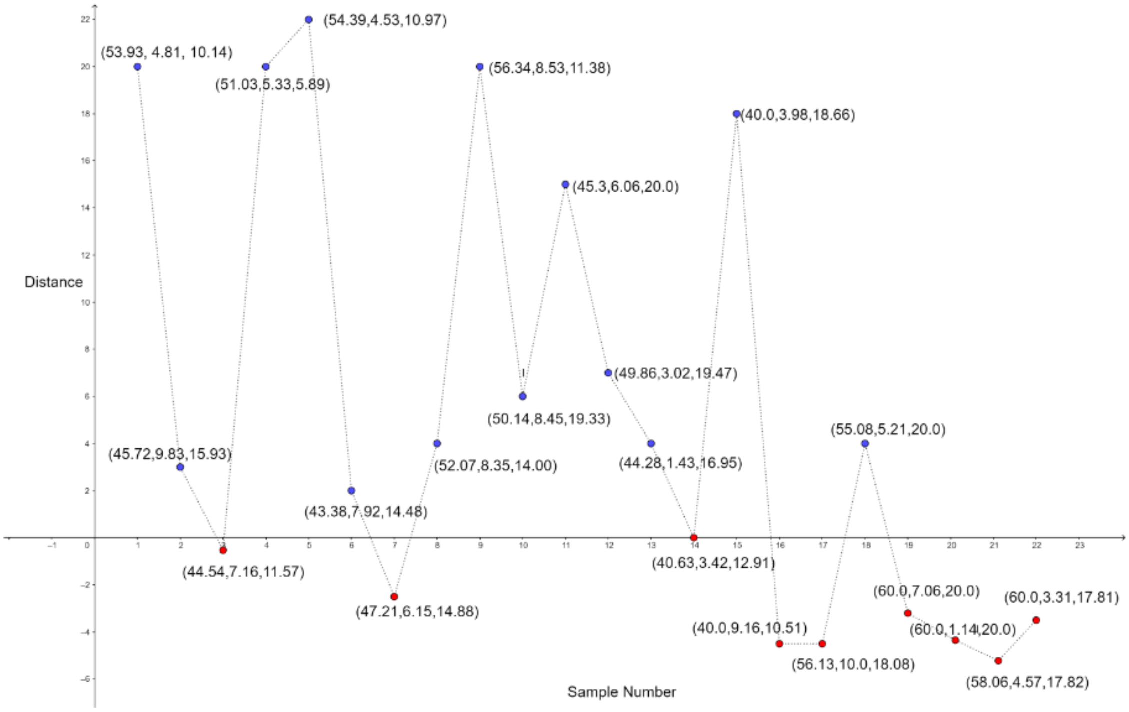


Figure 25: Plot illustrating function value of each sampled point up to 22 samples.

For Figure 25 above the deceleration for the plot is kept fixed at 3m/s2. Each point represents (subject vehicle speed, Pedestrian angle, Pedestrian speed). One of the left zero for the identified minima is found at [58.05,4.57°,15.69] and right zero at [58.05,4.57°,19.83].

The search space for this problem is extensive, with each point forming a test case. The step size λ is chosen as (domain range)/2 for each domain, which can be tuned for faster convergence. Multiple test cases were identified, such as [58.06, 4.57 °, 17.82, 3], [47.21, 6.15 °, 14.88, 3], [60, 0 °, 20, 4], [60, 10 °, 18.59, 3], [42.36, 7.19 °, 11.64, 4], etc. No violations were observed over a deceleration rate of 5 m/s², suggesting a safe driving strategy with a deceleration rate above 5 m/s².

The study presents a methodology for identifying test cases for complex black boxes like autonomous systems with fast convergence. A more detailed exploration of the method is available in the full paper [43]. This analysis can help define safe strategies for autonomous driving systems. Future work includes optimizing the algorithms by tuning hyper-parameters such as λ and searching new regions in parallel, as well as extending the methodology to identify parameter values for temporally changing functions and testing over more complex scenarios and use cases.

4.8 Surrogate Modelling with Gaussian Process Classifier

Within the SUNRISE project an approach was needed that allowed for a reduction in the number of tests conducted, while not needing previous knowledge of the scenarios, their parameters and the parameter distributions.

To achieve this goal, a method for selecting sampling points that focuses on two key aspects is proposed: exploitation, which targets parts of the parameter space known to be safety-critical, and exploration, which investigates regions where the existence of safety-critical scenarios is uncertain. Balancing these concepts of exploration and exploitation to minimize a function is well-understood within the optimization and machine learning domains. They are now adapted for use in scenario concretization and the SUNRISE Sub-Space Creation Methodology.

The input for this approach is a given logical scenario defined by a set of parameters and their ranges (scenario parameter space). Each concrete scenario allocated from the logical scenario is defined by a specific set of values for the free scenario parameters. Moreover, each concrete scenario is associated with a pass or fail outcome which is derived through simulation and calculation of given performance metrics, their combination into a scalar objective function, and the application of a given threshold γ. The selection of the appropriate metrics depends on the testing purpose and the system/function under test (SuT). The reader is referred to deliverable D5.3 [44] for an extensive discussion on metrics and their selection. The problem statement is described in

Figure 26, where the parameter space is presented in 2D for visualization purposes and the regions of pass and fail are highlighted.

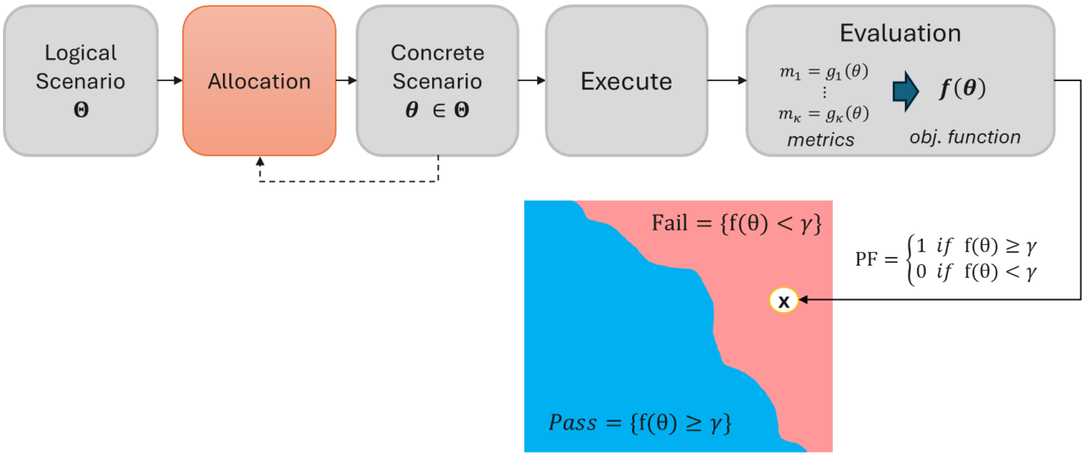

Logical Scenario $\boldsymbol{\Theta}$
Allocation
Concrete Scenario $\boldsymbol{\theta} \in \boldsymbol{\Theta}$
Execute
Evaluation
$m_1 = g_1(\theta)$
$\vdots$
$m_\kappa = g_\kappa(\theta)$
metrics
$\boldsymbol{f(\theta)}$
obj. function
$\text{Fail} = \{f(\theta) < \gamma\}$
$\text{PF} = \begin{cases} 1 & if\ f(\theta) \geq \gamma \\ 0 & if\ f(\theta) < \gamma \end{cases}$
x
$Pass = \{f(\theta) \geq \gamma\}$

Figure 26: Problem statement.

For this purpose, a Gaussian Process Classifier (GPC) is proposed to serve as a surrogate model with a two-fold role: i) to estimate safety evaluation outcomes (pass/fail) of given concrete scenarios without the need for execution (in simulation or proving grounds), hence allowing for the exhaustive assessment of the parameter space in feasible time and, ii) to assist in the sampling of uncertain concrete scenarios towards identification of the space boundary(ies), increase of the estimation confidence and parameter space coverage, and allocation of scenarios for further testing (e.g. from simulation to proving ground).

A GPC is a non-parametric machine learning model designed for classification tasks, where the goal is to predict discrete outcomes. Unlike traditional models that rely on a fixed set of parameters, a GPC defines a probability distribution over functions that could explain the observed data. This flexibility makes it well-suited for capturing complex and nonlinear relationships in input data, such as those influencing safety outcomes.

At the core of the GPC is the assumption of a latent function that links input features to classification outcomes. This latent function is not directly observed but is assumed to follow a Gaussian Process prior. A kernel function is used to express assumptions about similarity between data points, allowing the model to adapt to the structure of the data without requiring a predefined functional form.

Before any observations are made, the Gaussian Process (GP) considers a wide range of potential functions that could represent the data. As observations are collected, the model updates its belief, assigning higher probability to functions that better explain the observed outcomes. The result is a probability distribution over possible latent functions, which becomes more confident in regions where data is available.

The general pipeline for the GP-based surrogate model estimation is depicted in Figure 27, where the main components are depicted in orange:

- Conversion of estimated outcomes to class probabilities
- Initial sampling for training the GP
- Targeted sampling for selection of interesting scenarios for further training of the GP

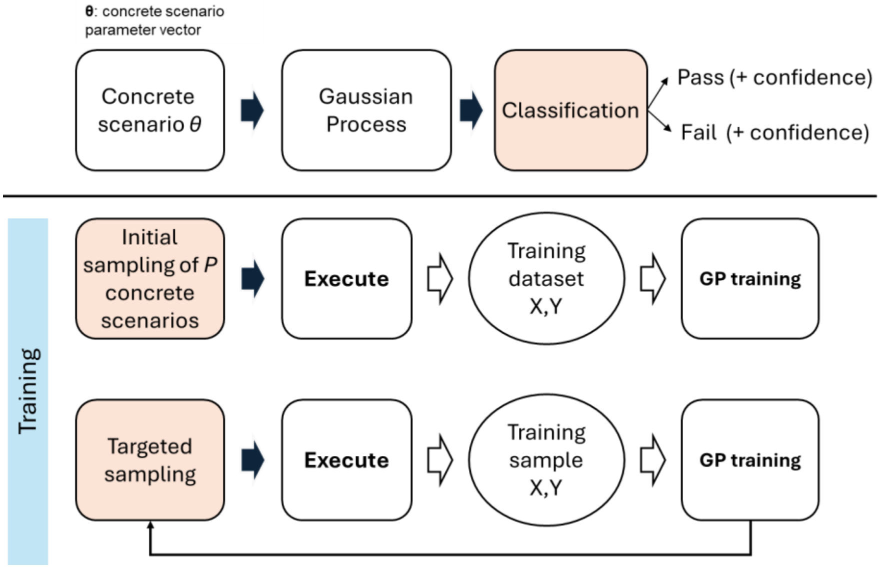


Figure 27: GPC estimation and training pipeline.

Different approaches may be taken to address the various components of this pipeline. In this document, two approaches will be presented as considered by different partners and are detailed below.

Approach 1:

To produce class probabilities, such as the probability of a scenario being unsafe, the GPC applies a link function, typically the *logistic function*. This maps the latent function values to the interval between 0 and 1, enabling probabilistic predictions that provide both a classification decision and a measure of confidence. Since the relationship between the latent function and class labels involves a non-Gaussian likelihood, exact inference is not feasible. Therefore, approximate inference is used, specifically the Laplace approximation, to estimate the posterior distribution over functions. This structure is ideal for binary safety evaluations, such as classifying scenarios as safe or unsafe. If the safety evaluation metric were continuous, a Gaussian Process Regressor (GPR) could be used instead. In that case, a link function would not be required because the output could be used directly.

The GPC is trained on a set of scenarios with known outcomes, allowing it to learn how input parameters influence the safety metric. Once trained, the GPC can be used to estimate the safety classification of new, unseen scenarios. This enables the approximation of the pass-fail boundary without simulating every possible case.

In the absence of prior knowledge, an initial set of scenarios is generated using LHS. After evaluating the initial scenarios and labelling the outcomes, the GPC is trained on this data. The resulting surrogate model captures the relationship between scenario parameters and the safety classification.

To further improve the model, Bayesian Optimization is used to select additional samples that are likely to provide the most informative results. Bayesian Optimization uses the surrogate model, in this case the GPC, to approximate the true safety evaluation function. The optimization process then applies an acquisition function to determine which scenarios to evaluate next. This function considers both the predicted safety metric and the uncertainty of the prediction, balancing exploration of uncertain areas with exploitation of promising ones. As a result, the optimization process efficiently identifies the pass-fail boundary.

The process follows the following structure in practice. It begins with the definition of relevant scenario parameters, their corresponding value ranges, and the result parameter, all of which are specified in a configuration file. Constraints can also be defined, for example, requiring that the target vehicle's speed is greater than that of the ego vehicle. These parameters can be derived either from the SUNRISE Data Framework or from expert knowledge. In addition to parameter specification, the number of initial samples to be generated is also defined. To ensure a near-random yet evenly distributed sampling of the parameter space, LHS is employed instead of traditional random generation. Each sample produced through LHS represents a unique test case.

Once generated, these test cases are executed within the SUNRISE simulation framework. Each simulation yields an outcome, classified as either a collision or a successful avoidance. The results, along with the corresponding input parameters, are then documented and passed to the Bayesian optimization module.

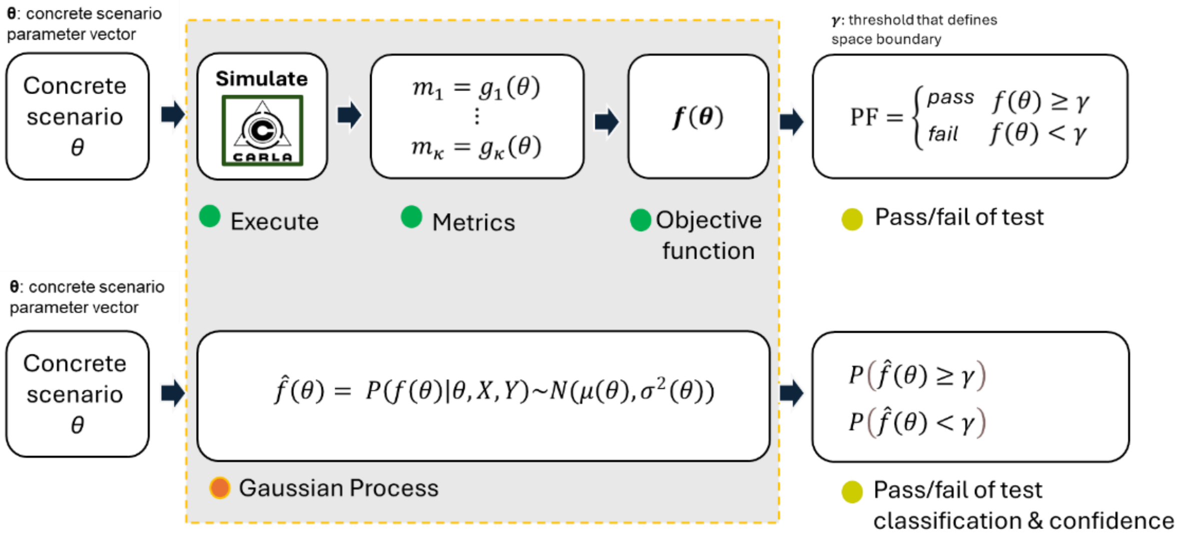
Within this module, a Gaussian Process (GP) classifier is trained on all available simulation data, including the most recent results. It learns to predict the outcomes of previously unseen scenarios, such as collisions or successful avoidances, while also estimating the confidence associated with each prediction. Based on these predictions, the Bayesian optimization identifies regions of the parameter space that are particularly informative, typically those characterized by low model confidence. To guide the sampling process effectively, the optimizer balances exploration, which involves random sampling across the parameter space, and exploitation, which focuses on the most informative regions. For exploitation, Bayesian optimization employs an acquisition function that leverages the GP's confidence estimates to guide the sampling strategy. New samples are then proposed in selected areas of both exploration and exploitation for evaluation in the next iteration.

In the next iteration, the proposed samples by the Bayesian optimization run through the SUNRISE simulation framework. The results are added to the existing simulation dataset, and the GP model is retrained accordingly. This cycle repeats iteratively, with each iteration improving the model's accuracy and reducing uncertainty. The optimization process continues until the confidence levels across the parameter space show no significant improvement, indicating convergence. To evaluate this, a set of equally spaced samples across the parameter space is fed into the GP model, which estimates the confidence levels for these samples. If the overall improvement in confidence is below a defined threshold, the stopping criterion is considered met, and the process terminates. This stopping criterion ensures that the iterative approach does not generate and simulate an excessive number of samples that contribute little new information, thereby avoiding unnecessary computational and simulation effort.

Approach 2:

The second approach is demonstrated in Figure 28, where the overall pipeline of the estimation based on the GPC surrogate model is shown. On the top we see the ground truth process in which a sample scenario is executed, metrics and outcome are calculated, and the pass/fail decision is made based on thresholding of the outcome with a given threshold γ. On the bottom we see the reciprocation of this process where the GP estimates the probability of an outcome as a normal distribution of the estimated mean and variance, and the pass/fail outcome is estimated by calculating the probability of the outcome being higher or lower than the threshold γ [. The GP kernel is used for the estimation of the mean and variance of the conditional posterior distribution $P(f(\theta)|\theta,X,Y)\sim N(\mu(\theta),\sigma^2(\theta))$, where θ is a new unseen sample (concrete scenario parameters), $f(\theta)$ is its estimated sample outcome (for the metric(s) chosen) , X is a set of samples and Y i

s their respective ground truth outcomes. To produce class probabilities, the probability of an outcome greater (or lower) than a given threshold γ is calculated by the cumulative normal distribution function $P(f(\theta)\geq\gamma)=CDF\left(\frac{\mu(\theta)-\gamma}{\sigma(\theta)}\right)$.



θ: concrete scenario parameter vector
Concrete scenario θ
Simulate
CARLA
$m_1 = g_1(\theta)$
$m_K = g_K(\theta)$
$f(\theta)$
γ: threshold that defines space boundary
$\mathrm{PF} = \begin{cases} pass & f(\theta) \geq \gamma \\ fail & f(\theta) < \gamma \end{cases}$
Execute
Metrics
Objective function
Pass/fail of test
θ: concrete scenario parameter vector
Concrete scenario θ
$\hat{f}(\theta) = P(f(\theta)|\theta, X, Y) \sim N(\mu(\theta), \sigma^2(\theta))$
$P(\hat{f}(\theta) \geq \gamma)$
$P(\hat{f}(\theta) < \gamma)$
Gaussian Process
Pass/fail of test classification & confidence

Figure 28: Pipeline of GPC concretization process.

In this approach, the first training phase includes the generation of training dataset through *random sampling* of concrete scenarios from the scenario parameter space and their pass/fail outcome is assessed via execution. The model is then batch-trained on the produced training set. In the second training phase, a guided sampling algorithm is employed to select concrete scenarios close to the pass/fail boundary. Each sampled scenario is subsequently executed, assessed as pass or fail, and added to the training dataset. The GP kernel parameters are updated every *K* iterations. The process is repeated until convergence or until a certain termination criterion is met (e.g. max number of iterations). For the guided sampling of scenarios close to the boundary, the *Straddle algorithm* is leveraged which employs a combined approach of exploration of points with high variance and exploitation of points close to the γ boundary.

Summary

The optimization process follows a structured loop:

1. Model the data: Train the GPC using the initial set of labeled scenarios that indicate safety or risk.

2. Quantify uncertainty: Use the GPC to predict outcomes for new scenarios and assess the model's confidence in those predictions.

3. Select new samples: Use an acquisition function, such as expected improvement, to select the next scenario to evaluate based on model predictions and uncertainty.

4. Evaluate and update: Simulate or test the selected scenario, label the result, and retrain the GPC. Repeat the process.

This iterative approach allows the surrogate model to improve continuously, resulting in a more accurate estimation of the safety evaluation boundary.

Indicative results

Approach 1:

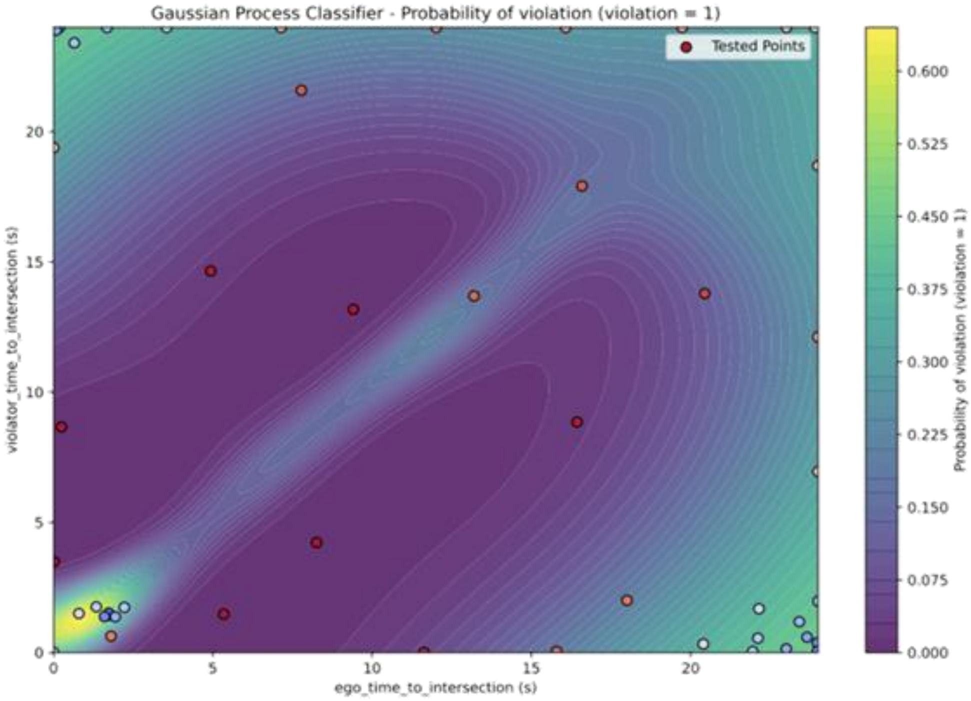


Figure 29: Estimation of probability of violation.

The graphic illustrates an example of this process using Use Case 1.2. In this scenario, the safety measure under consideration is a collision avoidance at an urban intersection, while the parameters defining the scenario space include the initial speed and distance from the intersection. The Bayesian Optimization process is employed to estimate the uncertainty associated with potential red-light violations across the parameter space. This enables the targeted selection of sample points that are most relevant for further testing of CCAM vehicles.

Approach 2:

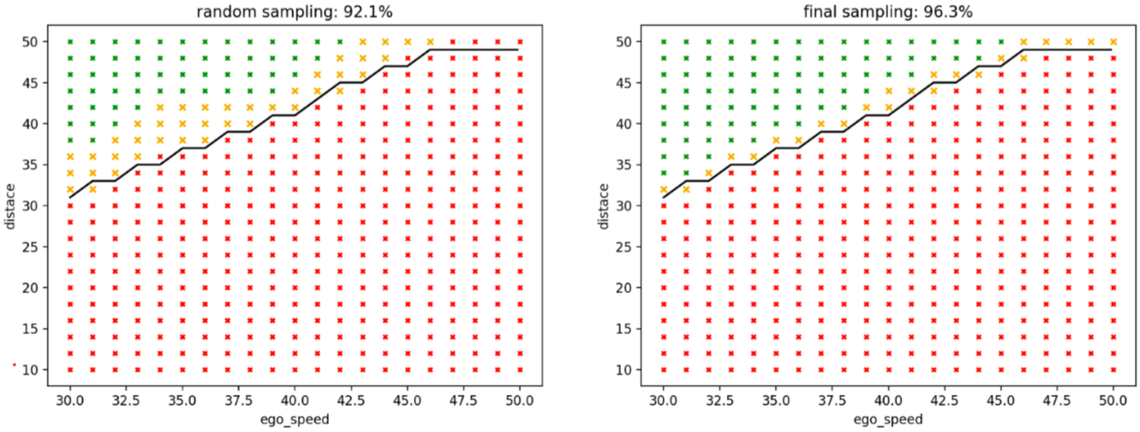


Figure 30: Estimation of uncertainty (yellow points) after the first (left) and second (right) training phases.

The above graphs illustrate the application of the proposed approach to Use Case 1.3 in which a pedestrian is darting out, initially occluded behind a static obstacle, while an ego vehicle is approaching. The free parameters of this logical scenario are the ego speed, the pedestrian speed and the distance between ego and pedestrian when the latter starts crossing. The two graphs depict the pass (green) and fail (red) estimations of the GPC of high confidence, while the low confidence (uncertain) estimations are depicted in yellow. On the left-hand side, the results are after the first training phase, while on the right-hand side, we can see the results after the second guided training phase. In both cases, we can see that the uncertain samples are those close to the pass-fail boundary. The number of uncertain samples is reduced after the second training phase.

5 PARAMETER SPACE COVERAGE

After a set of samples has been created in the parameter space of a logical scenario, it is very important to determine how well these scenarios cover the space. If too few samples have been chosen, the resulting assessment becomes unreliable, since failure regions might not be adequately detected within the space.

A detailed overview of coverage metrics, including metrics on database coverage, ODD coverage, as well as parameter coverage can be found in deliverable D5.3 [44] of the SUNRISE project. Coverage in the context of the safety assurance process itself is treated within the Coverage block of the SUNRISE SAF. It does however have a large influence on concretisation, especially for iterative approaches, as coverage is needed to determine the quality of the sampling. In this chapter, the focus lies on addressing parameter coverage in this context.

Coverage of a parameter space does not necessarily have to be uniform. In fact, this approach can be quite inefficient. Regions of the parameter space with low variability do not require many sample points. In contrast, areas with higher variability, such as those near the pass-fail boundary, need denser sampling. This is necessary to accurately estimate the behaviour of the evaluation metric across the scenario space.

5.1 Coverage in Discrete Spaces

In a discrete parameter space, the coverage definition becomes simple. All possible permutations of concrete scenarios are known. This means that a relation between all tested and all possible permutations can be easily established.

There might, however, still be a large number of possible permutations, so that it becomes sensible to further reduce them by focusing on parts of the space with higher variability, as discussed previously.

It is important to be able to estimate the coverage achieved to guide knowledge about the amount of tests needed. A Method for estimating coverage in discrete spaces by LHS approaches was presented by [45].The authors could derive an estimate for the probability of coverage. This probability is defined as:

$$P(k,n,d,t) = \frac{U(k,n,d,t)}{n^d}$$

Here, U denotes the number of cells, or parameter combinations, that have been tested with k trials of size n across a d-dimensional space, projected to a t-dimensional subspace. They could prove that this probability can be estimated by:

$$P(k,n,d,t) \approx 1 - e^{-k/n^{t-1}}$$

This estimation can be used to derive the necessary amount of samples for a LHS approach, depending on the coverage percentage that is desired.

A common challenge in testing is that the number of parameter combinations grows exponentially with the dimensionality of the space. In high-dimensional scenario parameter spaces, exhaustive testing becomes infeasible, even when the parameters are discretized. When full coverage is not possible, a metric is needed to assess how well the space is sampled. One such metric is t-wise coverage [46]. A test space satisfies t-wise coverage if all possible combinations of values for any *t* parameters are included at least once in the test matrix. For example, 2-wise coverage ensures that every pair of parameter values appears in at least one test case. Higher values of *t* improve coverage quality but require more test cases. This approach is based on the observation that most system faults are caused by interactions between only a small number of parameters.

5.2 Coverage in Continuous Spaces

Coverage in continuous spaces is quite a bit more complicated as than in discrete spaces. Since there is an infinite number of possible scenarios, the direct link between tested and possible scenarios is lost. As a first step, distance-based metrics between pairwise samples can be used to determine how much of the parameter space is touched by the samples. Commonly used pairwise distance metrics include the Euclidean and Mahalanobis distance among others. A Voronoi volume analysis can be performed to determine how uniformly a set of points covers a space. It does this by partitioning the space into regions around each sample point based on proximity. A Voronoi diagram is created to divide the space into cells, one for each cell, so that every point in a cell is closer to its corresponding sample than to any other sample. Large cells can indicate under sampled areas and low coverage.

This approach is insufficient, however. Large parts of the parameter space might be uninteresting to evaluate, for example because the criticality in this region might be low. In this region less evaluations might be desirable then closer to the pass-fail boundary. Therefore, it is quite desirable to define a methodology that allows for defining coverage in such a way that the regions with higher variability can be prioritized.

Traditionally in mathematical analysis Lipschitz continuity is used to evaluate the variability, or rate of change, of a function. A function is Lipschitz continuous if a real number, the Lipschitz constant K, exists that for every pair of points the absolute value of the slope between them is not greater than that number.

$$K \geq \frac{\left|f(x_i) - f\left(x_j\right)\right|}{\left\|x_i - x_j\right\|} \; \forall \left(x_i, x_j\right)$$

One can similarly compute a value K for all nearest neighbour sampling pairs and form a distribution over all resulting values. This can serve as a metric to evaluate coverage of the space. Higher values show higher variability and indicate insufficient coverage.

Another way to achieve this is the use of a surrogate model, like those already used in the sampling methodology described in section 4.8.

A surrogate model provides an approximation to the true function based on sampled data. But beyond point predictions, many surrogate models (especially probabilistic ones) can also provide estimates of uncertainty, meaning how confident the model is in its predictions at various points in the parameter space. These uncertainty estimates can serve as a proxy for coverage. Intuitively, if a model is very uncertain about a prediction at a point, it likely means the region around is sparsely sampled or exhibits high local variability, suggesting low coverage. Conversely, if the model is highly confident (low predictive uncertainty), the region is likely well covered. A powerful surrogate modelling approach is the Gaussian Process Regression (GPR). Gaussian Processes model a distribution over functions, meaning that instead of predicting a single fixed output for a given input, they define a probability distribution over all possible functions that could explain the observed data. This probabilistic formulation allows Gaussian Processes to express not just a best estimate, but a range of plausible outcomes.

For any new input point, a Gaussian Process provides a mean prediction, which represents the expected output value at that point based on the learned function. Simultaneously, it provides a standard deviation, which quantifies the model's uncertainty about that prediction. This uncertainty naturally increases in regions of the input space where little or no training data is available, and decreases in well-sampled areas, making it a valuable metric for identifying where the surrogate model is confident versus where the function is still largely unknown.

Mathematically this can be described as follows:

The start point is an unknown function $f(x)$ that maps all parameter combinations to their respective test results. This function can be evaluated for a certain parameter combination x_i^* by conducting a test (e.g. in simulation, on the proving ground). Now m different points $X^* = [x_1^*, x_2^*, \dots, x_m^*]$ have been evaluated giving $y^* = [y_1^*, y_2^*, \dots, y_m^*]$ where $y_i^* = f(x_i^*)$. The unknown function can be modelled as a Gaussian Process giving:

$$f(x) \sim GP\left(u_{pr}(x), k(x, x')\right)$$

With the prior mean function $u_{pr}(x)$, which is the best guess before seeing any data and the covariance function $k(x, x')$, which measures how related points x and x' are. This function is determined by a kernel.

Interestingly, one can now estimate the value of $f(x)$ at a new point without evaluating (testing) it. The joint prior distribution of the observed values y^* and the new unknown value y is:

$$\begin{bmatrix} y^* \\ y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} u_{\mathrm{pr}}(X^*) \\ u_{\mathrm{pr}}(x) \end{bmatrix}, \begin{bmatrix} K(X^*, X^*) & K(X^*, x) \\ K(x, X^*) & K(x, x) \end{bmatrix}\right)$$

With $K(A, B)$ being the covariance matrix with entries $k(a, b)\ \forall\ a \in A, b \in B$.

The posterior distribution, after seeing all the data, is still a Gaussian:

$$y \mid x, X^*, y^* \sim \mathcal{N}\left(u(x),\ \sigma^2(x)\right)$$

With the posterior mean:

$$u(x) = u_{\mathrm{pr}}(x) + K(x, X^*)K(X^*, X^*)^{-1}\left(y^* - u_{\mathrm{pr}}(X^*)\right)$$

And the posterior variance:

$$\sigma^2(x) = K(x,x) - K(x, X^*)K(X^*, X^*)^{-1}K(X^*, x)$$

The posterior mean is our resulting surrogate model. More information on the surrogate model can be found in section 4.8.

The posterior variance, however, is the uncertainty of the model at the new datapoint and our resulting coverage metric. In Figure 31 this uncertainty is visualised for an example use case, the assessment of a traffic light violation. It can be observed that uncertainty is highest at the pass-fail boundary in the scenario space. Testing at new points can reduce the uncertainty further, until a desired level of certainty over the entire space is reached.

More information on the mathematics behind the uncertainty quantification of Gaussian Processes can be found in [47].

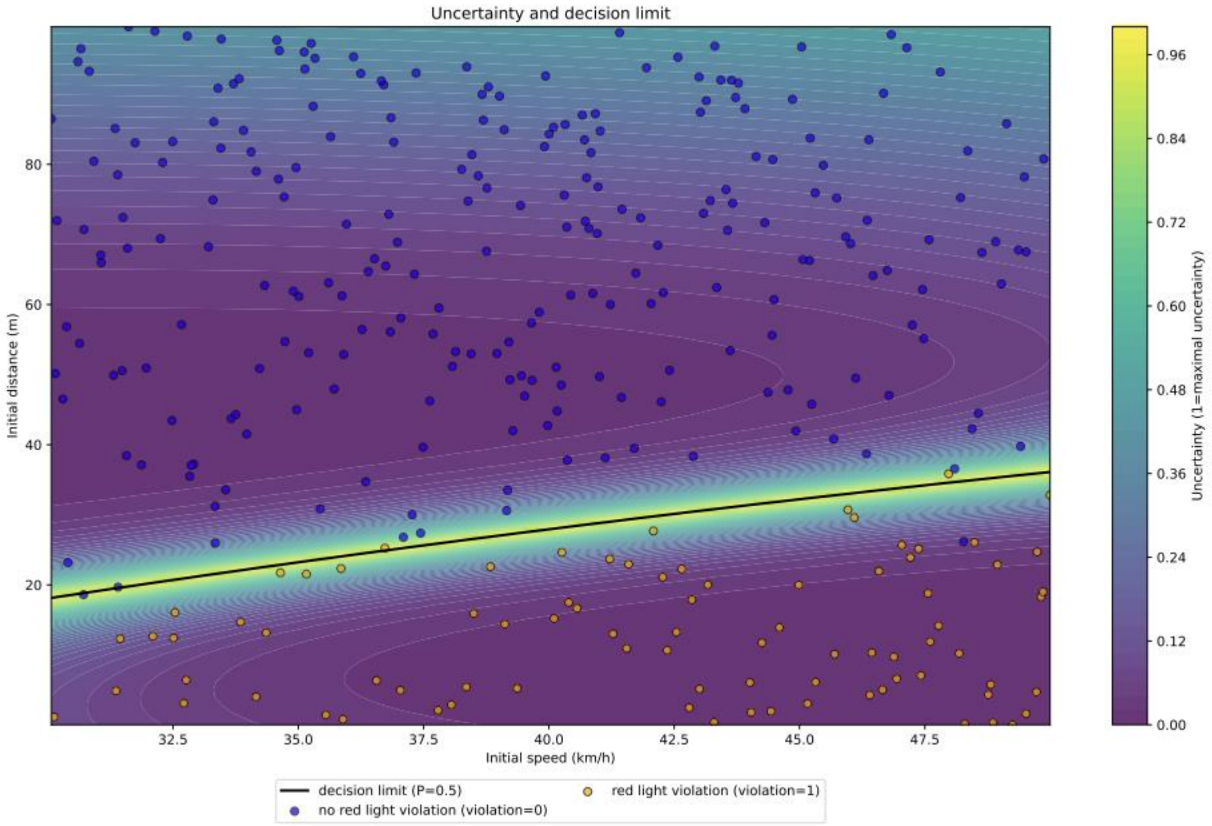


Figure 31: Representation of uncertainty in the parameter space.

A similar coverage criterion can be established even if uncertainty is not directly measurable. To assess this, the rate of change in the objective function can monitored over the different iterations. If the rate of change over multiple iterations is high the coverage is insufficient, as additional sample points provide new information to adjust the surrogate model. If the rate of change is low, no new information is provided.

This coverage estimation can be directly integrated into the sampling methodology, for which a stopping criterion is necessary to determine when the optimization process should end. This is done by first discretizing the parameter space into a grid. The size of this grid depends on the dimensionality of the parameter space and should be chosen based on available computational resources and coverage needs. Within each grid cell, a representative point is selected. This point can be chosen based on its position within the cell, as the average or mean of the evaluation metric over the cell, or using the approach described in section 4.8.

At each iteration, it is evaluated how the surrogate model’s prediction at these representative points changes. If the change in predictions falls below a predefined threshold, the model is considered to have converged. This indicates that the optimization process has sufficiently explored the parameter space and that the surrogate model has stabilized. As a result, the optimization can be stopped without the need to evaluate every possible scenario.

6 SAFETY EVALUATION METRICS

As CCAM technologies continue to evolve, ensuring the safety of such systems has become a critical priority. The deployment of CCAM solutions in real-world environments brings with it complex interactions among vehicles, infrastructure, pedestrians, and other road users. To manage these complexities effectively, safety metrics serve as a fundamental tool in assessing, monitoring, and improving the safety performance of automated systems.

This chapter presents a structured approach to identifying and applying safety metrics for the assessment of CCAM systems. These safety evaluation metrics serve as the foundation for the selection and sampling methodologies discussed in this deliverable. They are used to assess the performance of the CCAM system at various sample points, each representing a unique combination of parameters within the scenario space. The evaluation results derived from these metrics inform and guide the sampling strategy, helping to identify the parameter combinations necessary for a comprehensive and meaningful assessment of the CCAM system.

Safety evaluation metrics are an integral part of the Test Evaluate block within the SUNRISE SAF. However, they also play a crucial role during the concretization process. Many of the methods described in this deliverable use iterative test evaluations to guide sampling, refining the selection of test points to improve overall coverage and representativeness. To support this process, it is essential to understand which types of metrics can be applied in a safety assessment and how to define suitable reference criteria. These criteria provide the benchmark against which test results are compared, enabling a clear pass/fail decision for the CCAM system in a given test scenario.

6.1 Categories of Safety Metrics

Safety in CCAM systems must be assessed from multiple dimensions to ensure holistic and reliable performance. To achieve this, different types of safety metrics have to be considered in the safety assurance process, each capturing a different aspect of system behaviour, risk, and compliance. Note that not all metrics are applicable to all use cases, and the steps in the SAF may be applied multiple times for different metrics.

Some categories that might be considered in the safety assurance process are:

- **Criticality Metrics**, which evaluate the severity and urgency of situations.
- **Legal Compliance Metrics**, which ensure adherence to traffic rules and regulations.
- **Comfort metrics**, which reflect how humans may perceive the system’s safety performance.
- **Aggregated metrics**, which relate to the safety performance of the system over its entire ODD.

Together, these metrics form the foundation for assessing and validating the safety of CCAM solutions across design, testing, and deployment stages. An overview of a variety of metrics from different categories can be found in [48].

6.1.1 Criticality Metrics

Criticality metrics quantify how close a traffic situation is to becoming a potentially hazardous or unsafe event condition. These metrics are essential for evaluating the behaviour of automated driving systems under both normal and edge-case conditions. By identifying situations where the vehicle approaches safety thresholds, criticality metrics enable the detection of safety-relevant scenarios, support benchmarking against baseline performance, and help validate the effectiveness of risk mitigation strategies.

Unlike binary safety outcomes (e.g., crash or no crash), criticality metrics provide a more nuanced, and continuous assessment of risk. They allow stakeholders to understand not only whether the system avoided an incident, but also how safely it did so, and how much margin was available before intervention would have been required.

Criticality can be assessed using a variety of physical, temporal, and statistical indicators. These include spatial proximity to other road users, time-based measures of collision risk, and the likelihood of a situation escalating into an unsafe event. The examples in the following table illustrate some of the most commonly used criticality metrics.

A more complete overview of criticality metrics can be found in [49].

Table 3: Examples for Criticality Metrics.

Metric	Description	Example Use Case
Time-to-Collision (TTC)	Time remaining before a collision occurs if current speeds are maintained	Longitudinal risk assessment.
Time Headway (THW)	Time difference between vehicles in the same lane	Following distance evaluation.
Time-to-React (TTR)	Time available for the system to initiate a reaction	Emergency manoeuvre evaluation.
Post Encroachment Time (PET)	Describes the time gap between two vehicles at a point or area.	Evaluate criticality at intersections.

6.1.2 Legal Compliance Metrics

Legal compliance is essential for the safe and lawful operation of CCAM systems. As these vehicles are integrated into public road networks, they must consistently adhere to a broad spectrum of traffic regulations that govern human driving behaviour. Legal compliance metrics serve as a framework for evaluating whether automated systems conform to the rules of the road under a wide range of driving conditions and operational contexts. They should be derived from the respective road traffic regulation governing the ODD of the CCAM vehicle.

Fundamental metrics of legal compliance, depending on the ODD, can include obeying posted speed limits, adjusting speeds appropriately in variable speed zones, and adhering to specific limits in school zones, construction areas, or under adverse weather conditions. Compliance with right-of-way rules is equally important and entails yielding at intersections, giving precedence to pedestrians at crosswalks, and correctly navigating roundabouts and uncontrolled junctions correctly. Automated vehicles must also demonstrate reliable lane discipline, maintaining appropriate lateral positioning within marked lanes, executing legal lane changes with proper signalling and clearance, and avoiding illegal or unsafe driving manoeuvres such as lane straddling or shoulder usage.

Beyond these core behaviours, legal compliance extends to the interpretation and response to dynamic and static signage such as traffic lights, stop signs, yield signs, and temporary signals introduced by construction zones or law enforcement personnel. Vehicles must accurately detect and interpret these signs and signals and act upon them in a manner that is not only technically correct but legally compliant. In complex or congested environments, timely and proportional responses to signage and signals are critical to ensuring both safety and legality.

Additionally, these metrics should capture not only the frequency of compliance but also the quality and timeliness of actions taken in response to legal requirements. For instance, stopping at a red light must occur within the legal stopping zone and with sufficient lead time, while right-of-way decisions must reflect a correct understanding of local traffic laws.

Another critical area involves edge-case scenarios, situations where conventional traffic rules may not be clearly applicable or where infrastructure elements are missing or malfunctioning. Examples include unmarked roads where lane boundaries are undefined, intersections with non-functional traffic lights due to outages, or temporary construction zones that introduce unexpected patterns of movement. In such cases, the vehicle must demonstrate the ability to apply general legal principles and adopt conservative, safety-oriented behaviour while maintaining lawful operation. Assessing performance in these scenarios involves examining how well the system infers appropriate legal conduct and whether it remains within acceptable legal bounds despite the absence of explicit guidance.

Another emerging concern lies in inter-vehicle legal coordination, which becomes particularly relevant in scenarios involving merging, lane changes, or platooning. These manoeuvres require not only an understanding of static traffic rules but also dynamic interaction with surrounding vehicles, potentially through vehicle-to-vehicle (V2V) communication. Automated vehicles must evaluate right-of-way, maintain safe and legal following distances, and make coordinated decisions without violating traffic laws. Legal compliance in this context involves both individual behaviour and the ability to harmonize actions with others in a lawful manner. It is important to assess how successfully automated systems manage these shared responsibilities and whether any legal ambiguities arise from cooperative behaviour.

Lastly, operating in mixed traffic environments presents additional complexity, particularly when automated vehicles must interact with non-compliant or unpredictable road users. Pedestrians may jaywalk, cyclists might disregard signage, and human drivers can behave erratically or unlawfully. In such situations, the automated vehicle must maintain its own legal compliance while also adapting to ensure safety. The challenge lies in resolving these conflicts without committing legal infractions, such as avoiding an illegally crossing pedestrian without swerving into oncoming traffic or violating lane boundaries.

An example of evaluating legal compliance metrics in the context of the SUNRISE SAF, can be found in Use Case 1.2 of the SUNRISE project, detailed in deliverable D7.2 [4]. This Use Case deals with traffic light compliance.

6.1.3 Comfort Metrics

While comfort metrics are not directly tied to the primary safety performance of CCAM systems, they play a critical role in user acceptance and can indirectly influence overall safety outcomes. These metrics focus on evaluating how smoothly and predictably the automated driving system operates under typical driving conditions, especially from the perspective of the vehicle's occupants.

Key indicators include lateral and longitudinal acceleration, as well as jerk, the rate of change of acceleration and the interaction with other vehicles characterized by headways and relative velocities [48]. Although these parameters may remain within the vehicle's defined safety envelope and not pose a direct threat to physical safety, sudden or excessive changes in speed or direction can be perceived by passengers as uncomfortable, disorienting, unsafe or even alarming. For instance, abrupt braking, harsh cornering, or frequent small corrections in steering may degrade ride comfort even if they do not violate any traffic laws or mechanical limits.

Importantly, discomfort can have secondary safety implications. Low comfort levels may reduce user trust and acceptance of the automated driving function, potentially leading to disengagement or manual takeover in situations where the vehicle is actually operating safely. Such takeovers, especially if they are unplanned or reactionary, can introduce additional risks, as human drivers may not be fully aware of the system's status or the surrounding environment when resuming control. Therefore, maintaining a high standard of ride comfort contributes not only to user satisfaction but also to stable, uninterrupted system operation.

To comprehensively assess comfort, metrics should be collected across a variety of driving scenarios, including acceleration from a stop, deceleration during traffic flow changes, turning manoeuvres, and lane changes. These assessments help ensure that vehicle behaviour remains not only lawful and efficient but also intuitive and pleasant for passengers, reducing the likelihood of system disengagement and contributing to broader public acceptance of CCAM technologies.

The following table shows some comfort metrics identified by *[50]*. These metrics are vehicle dependent, meaning independent of human factors.

Table 4: Examples for vehicle related comfort metrics [50].

Metric	Description
Unclear Actions	Actions taken by the automated system that are unclear, unexpected, or unpredictable.
Driving Speed	Speed of the automated system.
Driving Headway	Headway of the automated system
Driving Action Execution	How the system undertakes certain driving actions, such as stopping, passing, turning, and accelerating.
Energy consumption	Energy (e.g. fuel) consumed by the automated system-

6.1.4 Aggregated Metrics

While the previous metrics are tailored to individual driving scenarios, it is equally important to assess the overall performance of a CCAM system across its entire ODD. This scenario-based approach aligns with the methodology used in the SAF, which emphasizes the evaluation of specific, well-defined situations. However, to gain a comprehensive understanding of system-level safety and reliability, certain metrics must be aggregated across a wide range of scenarios within the ODD.

Aggregated metrics can help to demonstrate the positive risk balance of the automated driving system. They provide evidence that, over time and across varied operational contexts, the system performs consistently, maintains safety, and delivers reduced risk compared to a human-driven baseline. These metrics are not limited to a single event or situation but instead reflect cumulative behaviour, making them essential for homologation and system validation.

The following table presents a set of example aggregated metrics that could be considered when evaluating CCAM systems across their full operational scope.

Table 5: Examples for aggregated metrics.

Metric	Description	Example Use Case
Crash Rate per 100km	Measures the number of crashes per 100 kilometres driven by the CCAM system.	Used to evaluate the overall safety performance of the system over time and benchmark against human drivers.
Emergency Manoeuvres per 1000km	Tracks the frequency of abrupt braking, swerving, or evasive actions triggered by the system to avoid a collision.	Helps identify how often the system approaches the limits of safe operation, even if crashes are avoided.
Take-over requests per 1000km	Counts the number of times the system requests a human driver to take control.	Assesses how often the system encounters situations outside its competence or ODD, requiring manual intervention.
Impact on Traffic Flow Efficiency	Measures the system's ability to maintain smooth and efficient integration into traffic (e.g., average speed, headway consistency, congestion contribution).	Used in urban or highway scenarios to assess whether the CCAM system contributes to or alleviates congestion.

6.2 Determining Safe Behaviour

Metrics can be used to assess the relative safety performance of a CCAM system. However, they do not in themselves define what constitutes safe behaviour. To make this determination, a reference is required.

A reference serves as a benchmark for interpreting safety metrics and can also be used to establish pass–fail criteria for specific scenarios. The choice of reference is crucial, as it sets the standard by which system safety is judged. For example, the reference might be:

- A collision-free requirement, where the system must avoid any collision that is physically avoidable.

- The behaviour of a competent and careful human driver.

In simulation environments, such references can be represented through driver models, computational constructs designed to simulate how a driver would act under specific conditions. These models are run on the same scenarios as the CCAM system, enabling direct performance comparison.

Driver models, in a broader sense, are mathematical or algorithmic systems that replicate how human drivers perceive their surroundings, make decisions, and control their vehicles. They range from simple rule-based models, which apply fixed responses to specific inputs, to advanced, data-driven models capable of closely mimicking human driving behaviour across diverse traffic situations by leveraging large amounts of data.

6.2.1 Idealized benchmark

Collision-free car following models are mathematical frameworks designed to ensure that a simulated or real vehicle can move through traffic without causing a crash, given reasonable assumptions about the behaviour of surrounding vehicles. They originated in traffic flow theory, where the goal was to realistically model vehicle interactions while guaranteeing safety constraints.

One of the earliest and most influential examples is the Gipps model [51]. Gipps introduced a car-following approach where each driver adjusts speed based on the vehicle ahead, maintaining a safe stopping distance under worst-case braking assumptions. The model incorporates limits on acceleration, desired speed, and reaction time, ensuring that a vehicle will never collide with the one in front if both follow the same rules.

The Intelligent Driver Model (IDM) [52] is another car following model. IDM calculates acceleration as a balance between a driver’s desire to reach a target speed and the need to keep a safe distance from the vehicle ahead. The “safe distance” is dynamic, depending on current speed, relative speed, and a minimum time gap, which makes the model adaptable to different traffic densities. Unlike the discrete safety margins in Gipps, IDM produces gradual accelerations and decelerations, making it more realistic.

However, these models result in behaviour that is not reflective of the capabilities of a human driver. This is especially true for challenging driving scenarios. Additionally, these models are only applicable for a limited number of scenarios involving car-following situations.

To address these limitations, frameworks extend the collision-free principle into more general reference models for CCAM systems. One prominent example is the Responsibility-Sensitive Safety (RSS) model [53]. RSS is a formal mathematical interpretation of the legal concept of Duty of Care, defining a set of rules that, if followed by all road users, would theoretically prevent accidents. These rules include:

1. Do not hit someone from behind.
2. Do not cut in recklessly.
3. Right-of-way is given, not taken.
4. Be cautious in areas with limited visibility.
5. If you can avoid an accident without causing another, you must do so.

RSS formalizes key notions such as safe distance, dangerous situations, and proper response for both longitudinal and lateral control. For example, in the simplest case of a single-lane road, the model defines the minimal safe longitudinal distance between two vehicles based on their speeds, braking and acceleration limits, and the following vehicle's reaction time. If the following car detects that the current distance is unsafe, it must react within a prescribed response time and then brake with at least a minimum deceleration until the danger is resolved.

Longitudinal safety is treated in two main cases:

- Same-direction travel – the following car must maintain enough distance to stop safely even if the lead car brakes hard after a reaction delay.
- Opposite-direction travel, both vehicles must brake, but the one going against the lane's intended direction is expected to brake harder.

Lateral safety is similarly defined: two vehicles must maintain enough side clearance to stop lateral motion toward each other before contact. RSS uses a robust μ-lateral velocity measure to account for small lateral oscillations that are not intentional manoeuvres.

A situation becomes dangerous only if it is unsafe both longitudinally and laterally. The danger threshold time marks when this condition first holds. RSS's proper response rules then constrain accelerations:

- If danger arises longitudinally, apply longitudinal braking rules.
- If danger arises laterally, apply lateral braking rules.

By design, these definitions are pairwise compatible: the ego vehicle can compute its safe acceleration limits for each other vehicle independently and then take the most restrictive limit, avoiding contradictory requirements.

This inductive safety principle allows RSS to guarantee that a properly responding vehicle will not cause a collision from behind, provided the other vehicle's braking does not exceed the assumed maximum. The model's parameters, reaction time, acceleration limits, and braking capabilities, can be tuned for different vehicle types or road conditions, enabling a trade-off between soundness (safety in more extreme situations) and usefulness (avoiding overly defensive driving that disrupts traffic flow).

Another similar model is the Safety Force Field (SFF) model developed by [54]. The is a general theory of safety for autonomous driving, designed to operate at the obstacle avoidance level. Its goal is to map the vehicle's perception of the world into a set of control constraints, limits on acceleration, braking, and steering, that, if obeyed, guarantee collision avoidance under the assumptions of the model.

The core idea is to treat every road user or obstacle as generating a “safety force field” around it, based on its current motion and the assumed safety procedure it can perform. For dynamic actors, the safety procedure typically means reducing lateral movement and braking to a stop as quickly as is reasonably possible; for static obstacles, it simply means remaining still. By predicting the trajectories that would result if each actor began its safety procedure immediately, SFF identifies space-time intersections between these trajectories.

The SFF approach is built on multiple key principles. First, it provides a collision prevention guarantee. That means, if all actors initiate their safety procedures before their predicted trajectories intersect, collisions cannot occur. Second, it supports better-than-baseline behaviour, allowing an actor to choose an alternative action, such as swerving instead of braking, that avoids a collision more effectively than the default safety procedure, provided it does not create new hazards for others. Third, it ensures clear responsibility, meaning that if an actor violates the safety force field and causes a potential intersection, this breach can be detected unambiguously.

In practice, SFF can be used to evaluate whether the vehicle’s planned actions respect all active safety force fields and identify unsafe actions. SFF simplifies validation and allows for consistent safety guarantees across different driving contexts.

6.2.2 Human driver reference models

The UNECE R157 regulation defines a performance model for Automated Lane Keeping Systems (ALKS) that uses the capabilities of a skilled and attentive human driver as a benchmark for determining whether traffic-critical situations are preventable or unpreventable [55]. The boundary between the two is based on simulation results using this human driver model, with the expectation that certain scenarios deemed unpreventable for humans may still be handled successfully by ALKS.

In low-speed scenarios, the model assumes that the human driver avoids collisions only through braking. The process is divided into three phases: Perception, where the driver detects a potential hazard; Decision, where the risk is evaluated and an action chosen; and Reaction, where braking is initiated and ramped up to full deceleration. Measured data informs the key timing parameters:

1. Risk perception time: 0.4 seconds.
2. Decision to braking initiation: 0.75 seconds.
3. Time to reach full deceleration: ~0.6 seconds to 0.774–0.85 g, depending on the scenario.

The model is applied to three primary ALKS scenarios:

- Cut-in – The system detects a vehicle moving into the ego vehicle’s lane when it crosses a lateral wander threshold of 0.375 m from the lane centre. The risk perception time starts at this moment, and the maximum lateral movement speed (from real-world data) determines the detection distance. A Time to Collision (TTC) of 2.0 seconds or less signals a high risk of collision.

- Cut-out – Detection also begins when the lead vehicle crosses the 0.375 m wander threshold while moving out of the lane. Here, the risk is defined by a Time Headway (THW) of 2.0 seconds or less, after which the ego vehicle must respond.
- Deceleration – Risk perception starts when the lead vehicle's deceleration exceeds 5 m/s^2.

To test these scenarios, the regulation specifies a set of environmental, roadway, and vehicle parameters: lane width, number of lanes, road grade, friction conditions, lighting, weather, vehicle speeds, initial distances, vehicle sizes, and dynamic motion data such as lateral velocity, maximum deceleration, and jerk rate. These parameters are varied in simulation to determine the exact combinations under which ALKS must avoid a collision at or below its maximum permitted operating speed.

In essence, the UNECE R157 performance model formalizes a human-equivalent baseline in perception, decision, and reaction capability, then applies it to realistic lane-keeping scenarios. ALKS is required to match or surpass this baseline in all preventable cases, ensuring that its performance meets or exceeds that of a skilled human driver.

Another similar model is the Fuzzy Safety Model [54]. The fuzzy safety model introduces two Fuzzy Surrogate Safety Metrics (SSMs) designed for real-time assessment of rear-end collision risk in both Advanced Driver Assistance Systems (ADAS) and automated vehicle functions. These are the Proactive Fuzzy Surrogate Safety Metric (PFS) and the Critical Fuzzy Surrogate Safety Metric (CFS). Their purpose is to overcome limitations of traditional safety metrics by using fuzzy logic to evaluate safety levels without relying on rigid, arbitrary thresholds.

Traditional SSM's, such as TTC, have been widely used for decades to estimate the likelihood of crashes based on vehicle trajectories. While useful, these metrics have notable limitations. They often overlook the influence of driver or system reaction time, which can be critical in real-world scenarios. Their binary classification of situations as either “safe” or “unsafe” relies on rigid thresholds that may not reflect the gradual nature of risk. Moreover, traditional SSMs focus on the proximity to a potential crash without capturing its possible severity, and the threshold values for metrics like TTC vary significantly across studies, making their application inconsistent.

Fuzzy logic addresses these problems by allowing gradual safety classifications, from very safe to very dangerous, rather than binary categories. This enables more nuanced, adaptive risk assessment, which is especially valuable for automated systems that must operate safely without being overly conservative.

The two metrics proposed are:

- PFS: Evaluates safety conditions before they become critical, allowing early interventions. It was compared against the safe distance definition of the Responsibility-Sensitive Safety (RSS) model and found to produce meaningful safety evaluations without fixed thresholds.

- CFS: Assesses risk when a situation has already become safety critical, outperforming TTC in predicting cases where vehicles failed to avoid a collision with a static obstacle.

The metrics were tested at the AstaZero proving ground in Sweden through two experiments:

- Car-following tests with five vehicles driven by Adaptive Cruise Control (ACC) to assess proactive metrics.

- Emergency braking tests against a static target to assess critical metrics, including measurements of maximum achievable deceleration.

Comparisons showed that CFS correlated strongly with collision severity (vehicle speed at impact) and outperformed TTC in early detection of unavoidable collisions. PFS provided a more flexible safety envelope than RSS, avoiding over-conservatism while still identifying unsafe conditions.

By integrating fuzzy logic into SSMs, this approach provides a robust and adaptable safety evaluation tool for both human-assisted and fully automated driving, better reflecting real-world variability and uncertainty.

A fundamental challenge in driving lies in tactical decision making, in contrast to the reaction to critical events the previous models captured, balancing the need to make progress toward one's destination with the need to avoid accidents, violations, and other negative consequences. While the primary purpose of driving is typically to arrive efficiently, this must be done under uncertainty, uncertainty about the behaviour of other vehicles, the sudden appearance of pedestrians, or changes in road conditions. Drivers who are overly cautious risk impeding traffic and behaving in ways that confuse other road users, whereas those who are overly assertive increase the likelihood of collisions. Human drivers generally excel at managing this trade-off, even in complex and unpredictable situations, which has made understanding their decision-making processes a key objective for researchers in traffic psychology, human factors, and autonomous vehicle development.

To address this problem, the active inference approach offers a promising framework that integrates goal-directed action and uncertainty management within a single computational process [56]. Originating in computational neuroscience, active inference models behaviour as the continual minimization of expected free energy (EFE). This measure combines two complementary components: pragmatic value, which drives the agent toward preferred outcomes (such as moving closer to the destination or avoiding hazards), and epistemic value, which quantifies the benefit of actions that reduce uncertainty about the environment (for example, checking the rearview mirror before overtaking). By representing both motives in a common in a single problem statement, active inference allows for an optimal balance between exploitation (progress) and exploration (information-seeking) to emerge naturally from the same decision-making process.

In practical terms, a driver modelled with active inference evaluates possible future action sequences, over a planning horizon and selects the one with the lowest expected free energy. This ensures that actions are chosen not only for their immediate utility in achieving goals but also for their potential to clarify uncertain aspects of the driving situation. For example, accelerating toward an occluding truck may have both pragmatic value (shortening travel time) and epistemic value (gaining a better view of the road ahead). Because the same mathematical formulation governs both types of value, the model inherently captures the way humans resolve uncertainty while pursuing their objectives.

Finally, cognitive models try to replicate the cognitive processes of the human drivers. These models generally give a broader applicability over more scenarios. A good example would be the SCM model [57]. The SCM is a cognitive driver model designed for motorway scenarios. It comprises six interconnected modules, Information Acquisition, Mental Model, Situation Manager, Action Manager, Action Implementation, and Driver Characteristics, that together simulate perception, cognition, decision-making, and action.

The Information Acquisition module models gaze allocation and fixation as stochastic processes, incorporating foveal and peripheral vision, the useful field of view, and tau theory for perceiving measures such as time headway and time to collision. Data from simulator and field studies inform these parameters. Perception is influenced by bottom-up (stimulus-driven) and top-down (task-driven) processes.

Perceived information is stored in the Mental Model, which holds microscopic (nearby vehicles), mesoscopic (aggregate traffic), and infrastructure data. This representation may be incomplete or error-prone but also supports calculations, such as assessing gap size for merging.

The Situation Manager classifies traffic scenarios based on extracted features (e.g., a turn signal indicating a lane change) and assigns situation intensities. The Action Manager then selects an appropriate response, braking, accelerating, or steering, while the Action Implementation module translates it into control inputs.

Driver Characteristics introduces behavioural diversity by assigning each simulated driver parameters for perception, rule compliance, and driving style, sampled from empirical distributions. This ensures realistic variability, so in the same situation, such as a cut-in, different drivers may respond with braking or lane changing.

A more complete overview of different models that might be used in the safety assurance process can be found in [58] and [59].

6.2.3 Using reference models in safety assurance

Previously, we described how the SUNRISE SAF can be used to evaluate the performance of a CCAM system in simulated, hybrid, or proving-ground tests. This process involves querying scenarios via the SUNRISE Data Framework, applying the concretization methods detailed in this deliverable, assigning the resulting concrete scenarios to test instances, and executing them.

The performance of a reference model can be assessed in much the same way as the CCAM system within a simulation environment. The most direct approach is to re-simulate every concrete scenario derived for the CCAM system using the reference model and then verify for each scenario that the CCAM system meets or exceeds the reference model's safety performance.

If evaluating the reference model for every concrete scenario is computationally prohibitive, a surrogate model can be employed to estimate the reference model's performance across the entire logical parameter space. The aim is to identify safe and unsafe regions, e.g. where the reference model does or does not result in a crash. These regions can then serve as pass–fail criteria: the CCAM system should not exhibit unsafe behaviour (such as crashes) in regions deemed safe by the reference model.

Beyond performance assessment, a computational model can also be used to optimize the scenario sampling process. By predicting outcomes across the parameter space, it can highlight regions that are:

- Not challenging for the system, and thus unlikely to reveal weaknesses.
- Already failed by the reference model, making further testing unnecessary.

Focusing test resources on the remaining, most informative regions can significantly reduce the number of required tests, an especially valuable benefit when testing capacity is limited, while still maintaining good coverage of critical and safety-relevant scenarios.

7 CONCLUSIONS

This deliverable presents a comprehensive set of approaches for **the concretization of logical scenarios** obtained through queries to the SUNRISE Data Framework. These approaches enable the generation of concrete scenarios that can be used for testing within the SUNRISE SAF. By detailing the concretization process in the **Query & Concretize** block, the deliverable contributes a vital element to the overall **SUNRISE methodology**.

To support the concretization of logical scenarios, a **categorization of sampling and selection methods** has been introduced. Approaches from literature have been identified and discussed in Chapter 3. Different approaches are designed to accommodate a range of user needs and varying levels of available information about scenario parameters. The resulting **Concretization Categorisation Framework** organizes and categorizes different concretization strategies and defines the relationships between them, providing a structured way to select and apply suitable methods based on specific testing requirements.

Further developments of existing approaches, as well as the introduction of new approaches have been detailed in Chapter 4. The focus of these developments has been the introduction of methods to allow for **Parameter Subspace Creation**, meaning that the testing efforts can be **focused** on certain relevant parts of the logical scenario **parameter space**.

To achieve this a key focus of this deliverable is the development **of iterative sampling methods** based on **surrogate models**. These methods support efficient identification of critical scenarios, particularly those near **pass-fail boundaries**. The surrogate model, which serves as an approximation of the evaluation metric across the scenario parameter space, is continuously improved using data from previous iterations. This results in an increasingly targeted and effective sampling process.

The concrete scenarios produced through these methods can be executed using other blocks of the **SUNRISE SAF** within its **Environment component**, including tools and processes defined in related deliverables. These include the **test allocation method** from Deliverable D3.3 and the execution procedures within the **Harmonized V&V Simulation Framework** developed within WP4. In addition, the methods described here provide valuable guidance for users of the SUNRISE SAF, helping them select appropriate concretization strategies based on the specific characteristics of the scenarios obtained from the SUNRISE Data Framework. They are especially relevant to **demonstrate the use cases** of demonstrated within WP7.

Additionally, this deliverable touches on important parts of the SUNRISE methodology that are relevant for the concretization of scenarios, including **parameter space coverage** in Chapter 5, which is also treated in WP5 of the project, and **safety evaluation metrics** for test evaluation in Chapter 6.

In conclusion, this deliverable provides information to support the successful concretization of logical scenarios for testing within the SUNRISE SAF. By enabling the identification of **parameter subspaces** where more or less testing is required, it contributes to a more efficient and focused concretization, showing that selected test cases within these parameter subspaces are *sufficient* instead of testing the entire parameter spectrum.

8 REFERENCES

[1] E. de Gelder, E. Kaynar, R. Ferreira, I. Panagiotopoulos, D. Becker, J. Beckmann, M. Skoglund, E. van Hassel, S. van Montfort, F. Hadj Selem, G. Ben Nejma, M. Nieto Doncel, A. Bruto da Costa, J. Zhang, M. Kirchengast und P. Weißensteiner, „D3.2 Report on Requirements on Scenario Concepts, Parameters and Attributes," 2025.

[2] X. Zhang, T. Menzel, A. Thorsén, B. Hillbrand, M. S. Ali und J. Beckmann, „D3.5 Report on the Dynamic Allocation and Validation of Test Runs," 2025.

[3] B. Hillbrand, M. Kirchengast, A. Balis, I. Panagiotopoulos, T. Menzel, T. Amler, E. Collado, J. Beckmann, M. Skoglund, A. Thorsén, T. Singh, M. S. Ali und M. Nieto, „D3.3 Report on the Initial Allocation of Scenarios to Test Instances," 2025.

[4] B. Hillbrand, G. B. Weiss, D. Hassler, S. Yetkin, B. Bourauel, J. M. Torres Camara, C. Berger, G. Villalonga, I. Panagiotopoulos, A. Bolovinou, E. Daskalaki, E. Collado, M. Miranda, T. Menzel, T. Amler, G. Stettinger, J. Beckmann, F. Navas, M. Skoglund, A. Thorsén, B. Sangchoolie, J. Uittenbogaard, M. Da Lio, P. Irvine, M. S. Ali, M. C. Rahal und G. Sayari, „D7.2 Safety Assurance Framework Demonstration Instances Design," 2024.

[5] T. Menzel, G. Bagschik und M. Maurer, „Scenarios for Development, Test and Validation of Automated Vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Changshu, 2018.

[6] M. Scholtes, L. Westhofen, L. R. Turner, K. Lotto, M. Schuldes, H. Weber, N. Wagener, C. Neurohr, M. H. Bollmann, F. Körtke und others, „6-layer model for a structured description and categorization of urban traffic and environment," *iEEE Access,* Bd. 9, p. 59131–59147, 2021.

[7] X. Zhang, J. Tao, K. Tan, M. Törngren, J. M. G. Sánchez, M. R. Ramli, X. Tao, M. Gyllenhammar, F. Wotawa, N. Mohan, M. Nica und H. Felbinger, „Finding critical scenarios for automated driving systems: A systematic literature review," *arXiv preprint arXiv:2110.08664,* 2021.

[8] S. K. Ahmed, „How to choose a sampling technique and determine sample size for research: A simplified guide for researchers," *Oral Oncology Reports,* Bd. 12, p. 100662, 2024.

[9] R. Schmidt, M. Voigt und R. Mailach, „Latin hypercube sampling-based Monte Carlo simulation: extension of the sample size and correlation control," in *Uncertainty Management for Robust Industrial Design in Aeronautics: Findings and Best Practice Collected During UMRIDA, a Collaborative Research Project (2013–2016) Funded by the European Union*, Cham, Springer International Publishing, 2018, p. 279–289.

[10] S. Raychaudhuri, „Introduction to Monte Carlo simulation," in *2008 Winter Simulation Conference*, 2008.

[11] C. P. Robert und G. Casella, „Metropolis–Hastings Algorithms," in *Introducing Monte Carlo Methods with R*, New, York: Springer New York, 2009, p. 167–197.

[12] A. E. Gelfand, „Gibbs Sampling," *Journal of the American Statistical Association,* Bd. 95, p. 1300–1304, 2000.

[13] J. S. Speagle, „A conceptual introduction to Markov chain Monte Carlo methods," *arXiv preprint arXiv:1909.12313,* 2019.

[14] E. de Gelder, H. Elrofai, A. Khabbaz Saberi, O. Op den Camp, J.-P. Paardekooper und B. De Schutter, „Risk Quantification for Automated Driving Systems in Real-World Driving Scenarios," *IEEE Access,* Bd. 9, p. 168953–168970, 2021.

[15] Y.-C. Chen, „A Tutorial on Kernel Density Estimation and Recent Advances," *Biostatistics & Epidemiology,* Bd. 1, p. 161–187, 2017.

[16] E. de Gelder, E. Cator, J.-P. Paardekooper, O. Op den Camp und B. De Schutter, „Constrained Sampling from a Kernel Density Estimator to Generate Scenarios for the Assessment of Automated Vehicles," in *IEEE Intelligent Vehicles Symposium Workshops (IV Workshop)*, 2021.

[17] B. W. Silverman, Density Estimation for Statistics and Data Analysis, CRC press, 1986.

[18] B. A. Turlach, „Bandwidth Selection in Kernel Density Estimation: A Review," 1993.

[19] E. G. Tabak und C. V. Turner, „A Family of Nonparametric Density Estimation Algorithms," *Communications on Pure and Applied Mathematics,* Bd. 66, p. 145–164, 2013.

[20] T. Berns, „Improving Scenario-Based Assessment of Automated Vehicles Using Event Data," 2023.

[21] E. de Gelder, M. Buermann und O. Op den Camp, „Comparing Normalizing Flows with Kernel Density Estimation in Estimating Risk of Automated Driving Systems," in *Proceedings of the IEEE International Automated Vehicle Validation Conference*, 2025.

[22] X. Zhang, J. Tao, K. Tan, M. Törngren, J. M. G. Sánchez, M. R. Ramli, X. Tao, M. Gyllenhammar, F. Wotawa, N. Mohan, M. Nica und H. Felbinger, „Finding Critical Scenarios for Automated Driving Systems: A Systematic Mapping Study," *IEEE Transactions on Software Engineering,* Bd. 49, Nr. 3, pp. 991-1026, March 2023.

[23] S. Hallerbach, Y. Xia, U. Eberle und F. Köster, „Simulation-Based Identification of Critical Scenarios for Cooperative and Automated Vehicles," in *2018 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2018.

[24] P. Junietz, F. Bonakdar, B. Klamann und H. Winner, „Criticality Metric for the Safety Validation of Automated Driving using Model Predictive Trajectory Optimization," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[25] T. Ponn, D. Koller und H. Winner, „A Method for Scenario Database Reduction by Clustering for Continuous Validation of Highly Automated Driving," *SAE International Journal of Connected and Automated Vehicles,* Bd. 3, p. 25–33, 2020.

[26] F. Hadj Selem, B. N. Ghada, K. Walid, D. Laurent und A. Emmanuel, „Which Scenarios Must Be Tested for Safety in Automated Driving: To Cut Testing Budget," in *Proceedings of the ITS World Congress*, Dubai, United Arab Emirates, 2024.

[27] M. Klischat und M. Althoff, „Generating Critical Test Scenarios for Automated Vehicles with Evolutionary Algorithms," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, Paris, 2019.

[28] T. Ponn, F. Müller und F. Diermeyer, „Systematic Analysis of the Sensor Coverage of Automated Vehicles Using Phenomenological Sensor Models," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, Paris, 2019.

[29] M. Koren, S. Alsaif, R. Lee und M. J. Kochenderfer, „Adaptive Stress Testing for Autonomous Vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Changshu, 2018.

[30] H. Beglerovic, M. Stolz und M. Horn, „Testing of autonomous vehicles using surrogate models and stochastic optimization," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Yokohama, 2017.

[31] C. E. Tuncali, T. P. Pavlic und G. Fainekos, „Utilizing S-TaLiRo as an Automatic Test Generation Framework for Autonomous Vehicles," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Rio de Janeiro, Brazil, 2016.

[32] E. de Gelder und J.-P. Paardekooper, „Assessment of Automated Driving Systems Using Real-Life Scenarios," in *IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[33] W. Ding, C. Xu, M. Arief, H.-C. Lin, B. Li und D. Zhao, „A Survey on Safety-Critical Driving Scenario Generation – A Methodological Perspective,“ *IEEE Transactions on Intelligent Transportation Systems,* Bd. 24, 2023.

[34] J. Langner, J. Bach, L. Ries, S. Otten, M. Holzäpfel und E. Sax, „Estimating the Uniqueness of Test Scenarios derived from Recorded Real-World-Driving-Data using Autoencoders,“ in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Changshu, 2018.

[35] R. Krajewski, T. Moers, D. Nerger und L. Eckstein, „Data-Driven Maneuver Modeling using Generative Adversarial Networks and Variational Autoencoders for Safety Validation of Highly Automated Vehicles,“ in *2018 IEEE 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, 2018.

[36] W. Chen und L. Kloul, „An Ontology-based Approach to Generate the Advanced Driver Assistance Use Cases of Highway Traffic,“ in *Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, Seville, 2018.

[37] Y. Li, J. Tao und F. Wotawa, „Ontology-based test generation for automated and autonomous driving functions,“ *Information and Software Technology,* Bd. 117, p. 106200, 2020.

[38] G. Bagschik, T. Menzel und M. Maurer, „Ontology based Scene Creation for the Development of Automated Vehicles,“ in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Changshu, 2018.

[39] M. Kirchengast, G. B. Weiß, B. Hillbrand und S. Solmaz, „Scenario-Based Testing of Automated Driving Functions Using Sampling and Iterative Optimization in a Modular Framework,“ in *Proceedings of the IEEE IAVVC 2025*, 2025.

[40] T. de Borba, „Methodology for Automated Definition of Critical Scenarios,“ 2019.

[41] V. K. Sundaramoorthy, „Identification of Interface Scenarios between Critical and Non-critical Driving Scenarios in n-D Parameter Space,“ 2021.

[42] F. H. Selem, G. B. Nejma, W. Kheriji, L. Durville, M. C. Rahal, S. Geronimi und E. Arnoux, „Developing a Methodology to Assess Data Completeness of Driving Scenarios for Testing Autonomous Vehicles: A Focus on ODD-Specific Objectives,“ in *Proceedings of the Driving Simulation Conference 2024 Europe VR*, Strasbourg, 2024.

[43] B. Gangopadhyay, S. Khastgir, S. Dey, P. Dasgupta, G. Montana und P. Jennings, „Identification of Test Cases for Automated Driving Systems Using Bayesian Optimization,“ in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Auckland, 2019.

[44] E. de Gelder, E. Kaynar, E. Mungan, H. Schneider, A. Bolovinou, E. Daskalaki, F. Hadj Selem, S. Vidal, S. Tarlowski, J. X. Zhang, J. Jeyachandran, T. Singh, M. Alirezaei, M. Nieto Doncel und Y. Taghipour Azar, „D5.3 Quality Metrics for Scenario Database Content,“ 2025.

[45] D. Donovan, K. Burrage, P. Burrage, T. A. McCourt, B. Thompson und E. Ş. Yazici, „Estimates of the coverage of parameter space by Latin Hypercube and Orthogonal Array-based sampling,“ *Applied Mathematical Modelling,* Bd. 57, p. 553–564, 2018.

[46] C. Amersbach und H. Winner, „Defining Required and Feasible Test Coverage for Scenario-Based Validation of Highly Automated Vehicles,“ in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019.

[47] J. Li und H. Wang, „Gaussian Processes Regression for Uncertainty Quantification: An Introductory Tutorial,“ *arXiv preprint arXiv:2502.03090,* 2025.

[48] M. Sharath und B. Mehran, „A Literature Review of Performance Metrics of Automated Driving Systems for On-Road Vehicles,“ *Frontiers in Future Transportation,* Bd. 2, November 2021.

[49] L. Westhofen, C. Neurohr, T. Koopmann, M. Butz, B. Schütt, F. Utesch, B. Neurohr, C. Gutenkunst und E. Böde, „Criticality metrics for automated driving: A review and

suitability analysis of the state of the art," *Archives of Computational Methods in Engineering,* Bd. 30, p. 1–35, 2023.

[50] V. Domova, R. M. Currano und D. Sirkin, „Comfort in automated driving: A literature survey and a high-level integrative framework," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies,* Bd. 8, p. 1–23, 2024.

[51] P. G. Gipps, „A behavioural car-following model for computer simulation," *Transportation Research Part B: Methodological,* Bd. 15, p. 105–111, 1981.

[52] M. Treiber, A. Hennecke und D. Helbing, „Congested traffic states in empirical observations and microscopic simulations," *Physical Review E,* Bd. 62, p. 1805, 2000.

[53] S. Shalev-Shwartz, S. Shammah und A. Shashua, „On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374,* 2017.

[54] K. Mattas, M. Makridis, G. Botzoris, A. Kriston, F. Minarini, B. Papadopoulos, F. Re, G. Rognelund und B. Ciuffo, „Fuzzy Surrogate Safety Metrics for real-time assessment of rear-end collision risk. A study based on empirical observations," *Accident Analysis & Prevention,* Bd. 148, p. 105794, 2020.

[55] United Nations Economic Commission for Europe, *UN Regulation No. 157: Uniform provisions concerning the approval of vehicles with regard to Automated Lane Keeping Systems (ALKS),* 2021.

[56] J. Engström, R. Wei, A. D. McDonald, A. Garcia, M. O'Kelly und L. Johnson, „Resolving uncertainty on the fly: modeling adaptive driving behavior as active inference," *Frontiers in Neurorobotics,* Bd. 18, p. 1341750, 2024.

[57] A. Fries, F. Fahrenkrog, K. Donauer, M. Mai und F. Raisch, „Driver behavior model for the safety assessment of automated driving," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022.

[58] J. Beckmann und J. Josten, Driver Performance Models as Reference for the Quality of Automated Driving Functions, fka GmbH in cooperation with the Institute for Automotive Engineering (ika), RWTH Aachen University, sponsored by the Research Association for Automotive Technology (FAT), 2025.

[59] C. Wang, F. Guo, R. Yu, L. Wang und Y. Zhang, „The application of driver models in the safety assessment of autonomous vehicles: Perspectives, insights, prospects," *IEEE Transactions on Intelligent Vehicles,* Bd. 9, p. 2364–2381, 2023.

[60] J. Tu, S. Suo, C. Zhang, K. Wong und R. Urtasun, „Towards Scalable Coverage-Based Testing of Autonomous Vehicles," in *Conference on Robot Learning*, 2023.

[61] I. M. Sobol, „Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates," *Mathematics and computers in simulation,* Bd. 55, p. 271–280, 2001.

[62] M. D. Morris, „Factorial sampling plans for preliminary computational experiments," *Technometrics,* Bd. 33, p. 161–174, 1991.

[63] D. Nistér, H.-L. Lee, J. Ng und Y. Wang, *The safety force field,* 2019.

[64] D. Zhao, „Accelerated Evaluation of Automated Vehicles," Ann Arbor, MI, USA, 2016.

[65] N. Weber, D. Frerichs, U. Eberle und others, „Safety-relevant Test Scenarios for Automated Driving Functions," *ATZ Worldwide,* Bd. 123, p. 52–57, 2021.

[66] L. Stark, M. Düring, S. Schoenawa, J. E. Maschke und C. M. Do, „Quantifying Vision Zero: Crash avoidance in rural and motorway accident scenarios by combination of ACC, AEB, and LKS projected to German accident occurrence," *Traffic Injury Prevention,* Bd. 20, p. 126–132, 2019.

[67] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick und F. Diermeyer, „Survey on Scenario-Based Safety Assessment of Automated Vehicles," *IEEE Access,* Bd. 8, p. 87456–87477, 2020.

[68] S. Li, W. Li, P. Li und others, „Novel Test Scenario Generation Technology for Performance Evaluation of Automated Vehicle," *International Journal of Automotive Technology,* Bd. 23, p. 1295–1312, 2022.

[69] J. Kapinski, J. V. Deshmukh, X. Jin, H. Ito und K. Butts, „Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques," *IEEE Control Systems Magazine,* Bd. 36, p. 45–64, December 2016.

[70] A. Gambi und G. Fraser, „Automatically Testing Self-Driving Cars with Search-Based Procedural Content Generation," *ACM Transactions on Software Engineering and Methodology,* Bd. 28, p. 1–26, 2019.

[71] D. Åsljung, J. Nilsson und J. Fredriksson, „Using Extreme Value Theory for Vehicle Level Safety Validation and Implications for Autonomous Vehicles," *IEEE Transactions on Intelligent Vehicles,* Bd. 2, p. 288–297, 2017.