IEEE Open Journal of
Intelligent Transportation
Systems

Received 14 July 2025; revised 5 January 2026; accepted 28 February 2026. Date of publication 9 March 2026; date of current version 24 March 2026.

Digital Object Identifier 10.1109/OJITS.2026.3670491

Toward Structured Training and Validation of AI-Based Systems With Digital Twin Scenarios

OSAMA MAQBOOL[1], ULRICH DAHMEN[1], MATHIAS TESCH[2], ANDRÉ KUPETZ[2], AND JÜRGEN ROßMANN[1]

[1]Institute of Man-Machine-Interaction, RWTH Aachen University, 52062 Aachen, Germany
[2]RIF e.V., 44227 Dortmund, Germany

CORRESPONDING AUTHOR: O. MAQBOOL (maqbool@mmi.rwth-aachen.de)

This work was supported in part by the Project "KImaDiZ" through German Aerospace Center (DLR) with funds of German Federal Ministry of Economics and Technology (BMWi) under Grant 50 RA 1934.

ABSTRACT Artificial intelligence (AI) has emerged as a pivotal technology for autonomous systems across various domains, but quality assurance remains challenging due to limited training data and inadequate validation methods. This paper presents a framework combining scenario engineering and digital twins to address these challenges. It supports AI system development from synthetic data generation to validation within a unified simulation-based approach. Scenario engineering focuses on real-world scenarios, while digital twins provide virtual replicas of system components throughout their lifecycle. The framework ensures flexibility in system design and testing, offering structured and transparent processes for quality assurance workflows. Two application case studies are presented: training a LiDAR-based pose estimation model in aerospace and validating an autonomous driving function. These case studies demonstrate the applicability of the framework and its potential for different phases of system development.

INDEX TERMS Scenario engineering, digital twins, simulation-assisted AI.

I. INTRODUCTION

ARTIFICIAL intelligence has developed within the last decade as the key technology for autonomous systems in domains ranging from automotive through production to aerospace. However, two major challenges present themselves in quality assurance of safety- and quality-critical applications.

First, good-quality training data is hard to come by. In areas such as industrial production or autonomous driving, data acquisition can incur high cost and safety risks. Aerospace missions represent an extreme case, wherein sensor data can only be generated during actual missions and real operating data might not be available at all during the development phase. In the automotive domain, comprehensive datasets such as Waymo Open Motion Dataset and nuScenes [1], [2] are available. However, they do not necessarily guarantee sufficient representation of maneuvers that are critical for specific features under development. Second, classical validation approaches are insufficient for reasonable assurance of system behavior. White-box testing methods have limited applicability to opaque machine learning components, and testing isolated AI-models may not be meaningful for the whole system [3], as the model interacts with various other components to exhibit collective system-level behavior.

This work proposes an integrated methodology and architectural framework that addresses these challenges through the symbiotic combination of scenario engineering and digital twins (DT) to create *digital twin scenarios*. The contribution is methodological: we integrate established simulation-based techniques within a unified framework that supports AI-based systems across their lifecycle, from targeted training data generation through systematic validation to design optimization.

The framework's applicability is demonstrated through two industrial case studies addressing specific stakeholder needs. The first focuses on precisely configured training data for a specialized aerospace pose estimation system. The second addresses standards-conformant validation of an automotive driving assistance module. To support integration with quality assurance standards, the framework provides a formalized implementation based on extensions of existing scenario specification languages. As the case studies demonstrate, this enables traceable workflows that facilitate collaboration between stakeholders.

The review of this article was arranged by Associate Editor Khaled Abdelghany.

© 2026 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

Scenario engineering shifts the focus from component-local tests to complete operational situations (*scenarios*) in which the entire system operates, considering higher-level parameters like environmental conditions or system interactions rather than low-level implementation details. This is complemented by a digital twin library that maintains virtual replicas of the system and its components throughout the lifecycle, enabling scenario engineering to flexibly compose and test different system configurations across diverse operational contexts.

The rest of the work is structured as follows: Section II presents the state of the art and identifies research gaps. Section III introduces digital twin scenarios. Section IV builds up on this to detail the scenario engineering process with digital twin scenarios. The two industrial case studies are detailed in Section VI, followed by conclusions in Section VII.

II. STATE OF THE ART

The section presents an overview of the state of the art for structured simulation-based approaches for training and validating AI-based systems. We will present the paradigms for synthetic training data generation, goals and methods for validation, and systematic processes that integrate these methods.

A. SYNTHETIC TRAINING DATA GENERATION

Within the domain of synthetic training data generation, the bulk of literature leans towards image generation for training computer vision models [4]. For multipurpose datasets however, simulations are necessary in which annotated virtual data from various sensors can be collected in different environments. The virtual KITTI [5] dataset uses real traffic scenes as seeds to create virtual test drives in Unity with varying domain shifts (including lightning conditions, camera angles, and weather conditions). The PreSIL dataset [6] provides high resolution RGB camera and LiDAR data extracted from the GTA-V game play. Although these approaches successfully achieve high fidelity in the data, the diversity of situations is not explicitly specified. The IDDA dataset [7] considers this problem and explicitly selects different towns, weather conditions, and sensor view points in data generation scenarios. In a similar vein, the SHIFT dataset [8] also considers different traffic and crowd densities, and features continuous changes across variation factors. In parallel, large-scale real-world datasets such as nuScenes [2] and Waymo Open Motion [1] provide extensive annotated driving data collected from operational fleets. These datasets have been instrumental for benchmarking and pretraining; however, their scenario composition and coverage are fixed by the data collection process and cannot be adapted to specific system configurations, operational design domains, or validation objectives. In general, these datasets are useful for generic pretraining of machine learning models. However, further training requires application-specific data with explicit coverage of critical situations. Systematic simulation approaches are therefore required which allow tailored data generation for particular system configurations (e.g. type of sensor), scenarios (e.g. degree of criticality) and fidelity.

B. VALIDATION METHODS FOR AI-BASED SYSTEMS

Validation of AI-based systems is often treated as a separate research domain. We classify simulation-based validation broadly into *coverage*, *optimization*, *uncertainty quantification* and *formal verification* methods.

Coverage-based methods aim to maximize the coverage of functional requirements or test parameters. Hauer et al. [9] propose probablistic reasoning to ensure that all test *types* have been covered. Discrete parameter coverage is evaluated through combinatorial techniques such as T-wise tests [10], which measures the proportion of unique T-parameter combinations covered during testing relative to the total number of possible T-parameter combinations. Another form of coverage is probabilistic coverage, which estimates the likelihood of a fault occurring within a given confidence interval. This can be achieved via Monte Carlo simulations [11]. However, since critical events are rare, importance sampling is often employed to increase the sampling rate of such events [12].

In contrast to coverage, optimization-based methods generally aim towards the *falsification*——finding counterexamples to safety or correctness specifications within a parameterized test space. Corso et al. [13] provide a taxonomy of commonly used methods, including simulated annealing, evolutionary algorithms and bayesian optimization as frequently used in this context. Tuncali et al. [14] present a hybrid approach that combines coverage and falsification: they first achieve baseline coverage using covering arrays, then use the most promising results to initialize a fault-search method.

In context of autonomous systems, uncertainty quantification is commonly addressed through statistical validation methods that aim to estimate the probability of hazardous events. Monte Carlo simulation and statistical model checking are widely used to derive probabilistic confidence bounds on safety-relevant properties over a given operational domain [11]. To address the rarity of critical events, importance sampling and rare-event simulation techniques have been proposed to efficiently estimate hazard probabilities such as collisions or safety violations [12].

Works in formal verification use formal specification and verification techniques to analyze safety-critical AI systems. Tools such as VerifAI [15] provide simulation-guided formal analysis capabilities, including temporal logic falsification, parameter synthesis, and counterexample diagnostics for systems with machine learning components. More recent work has extended formal approaches to compositional and probabilistic analyses that explicitly address perception subsystems within autonomous agents [16].

C. VALIDATION PROCESSES AND STANDARDS

The broad taxonomy of presented methods constitute the *quantitative* aspect of validation. For their effective

application to safety- and performance-critical systems, they must be integrated within a systematic testing workflows, that define *what* is tested, *under which* conditions and with respect to *which system configuration*. The automotive SOTIF (ISO/PAS 21448) standard [17] prescribes the identification of the operational design domain of safety-critical functions and the usage of scenarios for structured testing. The PEGASUS project [18] provides a road-map for this by decomposing scenarios into functional, logical and concrete scenarios. Both these frameworks effectively establish *what* is to be tested. Within this context, validation becomes the exploration of a parameterized scenario space, to which a wide range of quantitative methods—such as optimization, falsification, and active exploration—can be applied [19]. Safety or correctness objectives can also be formalized using temporal logic or metric-based specifications derived from system requirements [14]. These diverse methods built on top of systematic frameworks like SOTIF additionally establish *how* scenarios are explored, but do not cover how system models and configurations are represented, instantiated, and varied across testing activities. Most existing approaches provide limited support for explicitly modeling system artifacts—such as sensors, vehicle dynamics, or planners—in a way that allows validation results to be traced, reused, and transferred across development phases and application domains. This lack of bidirectional interaction between system modeling and testing has also been identified by the systems engineering community as an obstacle to continuous verification and validation of evolving systems [20], [21].

Complementary to these process-oriented works, domain-specific languages enable formal and probabilistic specifications of scenes and traffic situations. OpenSCENARIO [22] specifies a formal model according to which parameterized traffic situations can be specified in XML. Scenic [23] offers a more intuitive process by developing a domain specific language that allows the description of scenes with probabilistic distribution of elements (e.g., the positions of participants). It has been used with automotive simulators such as Carla [24] and robotic simulators such as Webots [25]. The OpenSCENARIO V2.0 standard [26] proposes a similar domain specific language for specifying probabilistic scenarios. Similar scenario-based testing approaches have been adopted in other safety-critical domains, such as railway systems, where scenario frameworks are employed to validate highly automated control systems [27].

D. RESEARCH GAP AND CONTRIBUTION

The identified gaps from state-of-the-art, crucial for supporting AI-based systems throughout their life-cycle, are summarized as follows.

- While large-scale (real or synthetic) datasets are available for pretraining machine learning models, there remains a lack of systematic frameworks for fine-tuning these models to specific applications. This gap is particularly critical in aerospace, where specialized sensors, inaccessible operational data, and mission-critical constraints require systematic yet domain-tailored training data generation strategies that are currently absent.
- Validation strategies, most prominently scenario-based strategies, lack explicit representation of system artifacts to be simulated. This can lead to inconsistent test outcomes across simulators and fidelity levels, and prevents integration of development models into the validation process. As a result, validation procedures cannot scale across the system lifecycle, and test results cannot be generalized across projects and application domains.
- Simulation-based training and validation are often handled as separate domains. There is a lack of unified frameworks that provide a consistent foundation for both, which is crucial for retraining systems based on validation outcomes.

To address these gaps, this work proposes a framework for structured generation and execution of DT-scenarios that supports both training and validation activities. Importantly, the contribution is not a new safety standard or scenario language. Rather, the framework presents an execution layer that complements existing standards such as SOTIF and can be realized by adapting different languages such as OpenSCENARIO or Scenic.

DT-Scenarios allows system design to be an explicit part of scenario engineering and enables quantitative exploration methods to be applied consistently across training, validation and system evolution. Similarly, the choice of simulator such as Carla is another complementary aspect. Different simulators can be used to implement our framework so long as they support multi-domain simulations and the modular architecture required for DT-Scenarios. This work uses the VEROSIM simulator [28] for this purpose, to which we have added scenario parsing and instantiation functionalities as part of this work.

While earlier works by the authors introduced foundational concepts—including a formal meta-model for scenario specification [29] and conceptual outlines of DT-based scenario workflows [30], [31]—those works were limited to theoretical formulations and validation-focused applications. Building upon these foundations, the present work contributes the following.

- A technical realization of the DT-scenario architecture and its integration with formalized scenario specifications for automated scenario generation.
- An extended methodology that encompasses both training and validation phases, including feedback mechanisms between validation results and model retraining.
- Two industrial case studies spanning aerospace and automotive domains that demonstrate practical applicability of the proposed approach.

III. ROLE OF DIGITAL TWINS

The presented approach uses DTs as a central structuring element for diverse activities in the design, training, verification

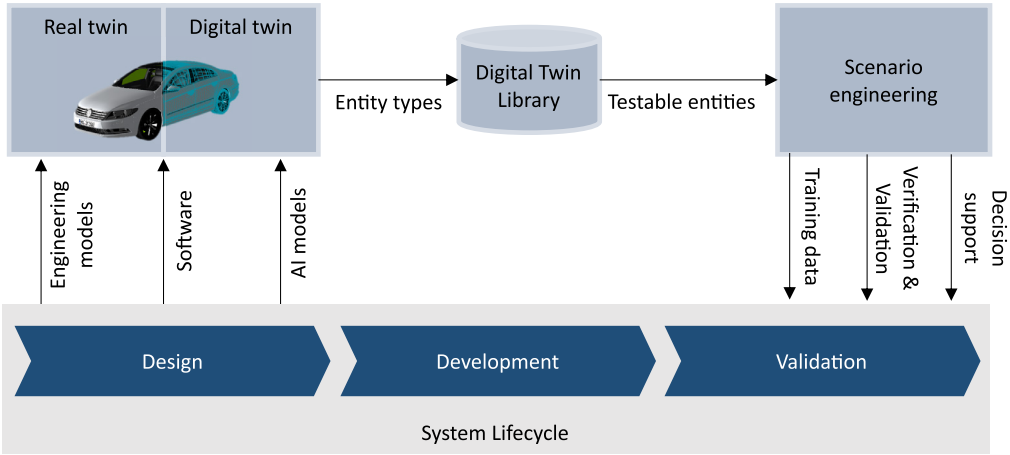


FIGURE 1. Symbiotic interaction between digital twins and scenario engineering throughout the system life-cycle.

and validation of the system of interest (SoI). The overall envisioned system-development process is illustrated in Figure 1. Artifacts created during engineering activities are actively integrated within the DT, the co-evolving virtual replica of the SoI. Sub-systems and variants of the SoI are thereby collected in a DT-library. In turn, scenario engineering (detailed in Section IV) can combine different *testable entities* to create precise variants of the SoI, which subject to diverse operational situations can deliver training data, verification and validation, and serve as a general decision support framework. The following sections detail the process.

A. DEFINITION OF A DIGITAL TWIN

Since the term digital twin (DT) can be used differently across application domains, we restrict its usage in this paper to the context of simulation models. Following Schluse's definition [32], the DT is a 1-to-1 virtual representation of the SoI, reflecting its structural, behavioral and interactive properties. It can integrate artifacts created during systems engineering and development, such as engineering models, software components, AI models, thereby co-evolving with the SoI. Moreover, DTs can be aggregated in hierarchical and heterarchial structures to create *system of systems*. The DT is simulated as a DT-scenario in a virtual testbed, a process detailed in following.

B. DIGITAL TWIN SCENARIOS

A *digital twin scenario* is an executable simulation model that captures the interactions between DTs, their components, and the environment. As illustrated in Figure 2, a typical DT-scenario for AI-based applications consists of three types of elements: the system of interest (e.g., an ego vehicle), adversaries (e.g., traffic participants), and the environment (e.g., a road network).

Each DT reflects the structure of its corresponding real-world counterpart and integrates multiple domain representations. These include CAD models, kinematic

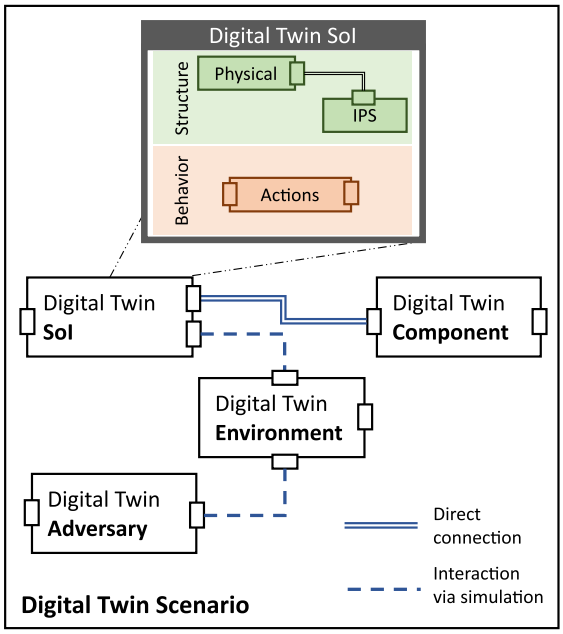


FIGURE 2. General structure of a DT-scenario comprising SoI, components, environment and adversaries. DTs can be added at the highest abstraction layer, or components from lower abstraction layers can be explicitly connected to respective systems. Actions of DTs are used to execute the *scenario story*, wherein DTs may interact with each other.

and dynamic models, specifications, and software artifacts. The virtual testbed executing the scenario provides domain-specific simulation algorithms such as dynamics, 3D rendering, sensor simulation, and software containers. This combination enables comprehensive cross-domain simulations.

DT-scenarios support flexible levels of abstraction. By default, DTs are modeled at the highest system level (e.g., complete vehicle, complete environment). However, specific lower-level components can be explicitly instantiated and connected when needed. This enables convenient variability: for instance, multiple traffic scenarios can be created where

the same ego vehicle DT is equipped with different sensor configurations to optimize perception system design.

Each DT implements controllable behaviors through *actions*—for example, a vehicle DT may offer lane-change or speed adjustment actions. These actions are orchestrated through a *simulation story*, which defines a sequence of maneuvers that trigger DT actions according to a specified schedule or event-based conditions. During execution, the virtual testbed ensures that appropriate interactions—such as physical effects like collisions and friction—are simulated correctly throughout the scenario.

This modular architecture provides two key dimensions of flexibility. First, the system design can be varied by selecting different DT components—enabling training data generation with specialized sensors or validation testing with integrated AI models. Second, the operational context can be precisely specified through the simulation story, allowing systematic exploration of diverse operational conditions. Together, these capabilities enable DT-scenarios to serve as a unified foundation for both training and validation activities throughout the system lifecycle.

C. ADVANTAGES OF THE DIGITAL TWIN APPROACH

The key advantages of DT-based simulations, particularly for AI-based systems, are highlighted below.

- They provide control over model- and simulation-fidelity. Model fidelity refers to the level of detail of components that make up the DT (e.g. ideal or physically realistic camera connected via ideal or protocol-specific connections), whereas simulation fidelity pertains to the accuracy of simulation algorithms employed by the virtual testbed. Training or validation data can therefore be consistently generated with the desired quality.
- Re-usability of models allows DTs from different stages of SoI life-cycle to be compared against intended operational conditions. This facilitates system-level regression tests, enabling the comparison and optimization of system behavior across different life-cycle phases.
- It integrates systems engineering activities directly within training and validation pipelines. As will be demonstrated in the next section, scenario engineering approaches do not work with idealized or obsolete SoI models, but with the latest available versions, with full control over the degree of system abstraction relevant for test scenarios.

The afore-mentioned potential of DTs is utilized by scenario engineering detailed in Section IV. The presented methods allow the formalized specification and automated generation of DT-scenarios via scenario engineering.

IV. SCENARIO ENGINEERING WORKFLOW

The previous section explored the potential of digital twin scenarios to provide flexibility along both system design and operational situations. Scenario engineering exploits this potential through automated generation and evaluation of DT-scenarios at scale. Building on the scenario abstraction levels from PEGASUS [18], the workflow extends this foundation in two key aspects: (1) explicit integration of the DT-library enables scenarios to incorporate and track evolving system artifacts throughout development, and (2) feedback mechanisms connect validation outcomes to model retraining through pluggable exploration strategies in the scenario generation loop.

The workflow comprises three main stages. First, *scenario design* produces large-scale formalized specifications of specific operational situations varying across key dimensions. Second, *simulation* instantiates and executes these concrete scenarios as DT-scenarios within virtual test-beds, generating training data or validation results. Third, *evaluation and exploration* analyzes simulation outcomes and guides further scenario generation, either for iterative model refinement during training or for systematic coverage during validation. The following subsections detail each stage.

A. SCENARIO DESIGN

The scenario design phase operationalizes the *operational design domain* (ODD) as a structured scenario space. The ODD includes the environmental, operational, and behavioral conditions under which the system is designed to function correctly [17]. For training, the ODD defines the diversity of situations the system should learn; for validation, it delineates where correct behavior must be verified.

The input to the scenario design phase is the DT-library as a repository of executable models which can be used to instantiate scenario entities. In turn, three types of scenario specifications are produced with increasing concreteness: an abstract scenario defining the scenario template with parameterizable elements, a logical scenario specifying the parameter space and variation constraints, and concrete scenarios representing specific instantiations ready for simulation.

B. ENTITY CATALOG

Prior to scenario design, the DT-library is organized into an entity catalog that indexes available models along with their interfaces and capabilities. Each catalog entry specifies the DT's controllable actions (e.g., lane change, acceleration) and connection points for integration with other components. This structured catalog enables scenario designers to systematically select and compose DTs when defining abstract scenarios.

1) ABSTRACT SCENARIO

The abstract scenario describes a single operational situation as a parameterized template within an XML-file following the meta-model in Figure 3. It comprises two main sections. The first is an *entity topology* that maps scenario entities to DT models from the entity catalog. The second is a *behavior description* that orchestrates the scenario story through structured sequences of entity actions. Key parameters within both sections are left undefined and explicitly marked for

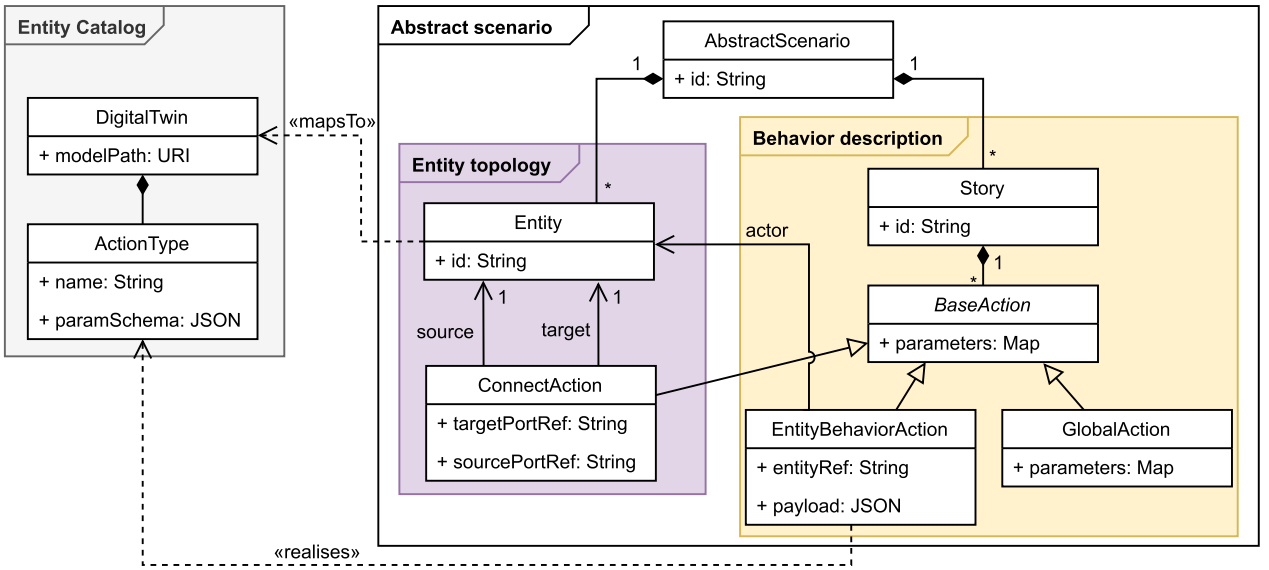


FIGURE 3. Meta-model for abstract scenario.

parameterization, enabling systematic variation across the scenario space.

The entity topology section declares scenario entities, i.e. participants of the scenario by mapping them to particular DT models in the entity catalog. In addition, a `ConnectAction` can be used to specify structural bindings between entities. These two elements can be used to describe the DT-scenario structure in Figure 2. In the simulation stage, each entity is instantiated by its corresponding DT and the entity connections are realized via communication interfaces between the instantiated components resulting in a composite DT.

The behavior description section follows the OpenSCENARIO [22] structure but deviates on certain aspects to allow entities from different domains to be incorporated. It organizes the scenario story through scenes, maneuver groups, and maneuvers that orchestrate event-triggered entity actions. Three types of actions are supported. First, the `ConnectAction` (specified during initialization) establishes connections between entities. Second, `GlobalActions` define generic activities such as waiting and environment-related effects like time of day and weather. Third, `EntityBehaviorActions` are dynamically actions which are defined within the entity catalog alongside each DT. For example, a vehicle DT declares its own `ActionTypes` such as lane change or speed adjustment, allowing the behavior description to invoke these DT-specific actions for entities mapped to that model. This ensures that new capabilities and domains can flexibly use the scenario formalism.

An example of automotive abstract scenario is illustrated in Figure 4, featuring entities for ego vehicle, npc vehicle and an ADAS component. The initialization stage specifies connection between the ego vehicle and ADAS component,

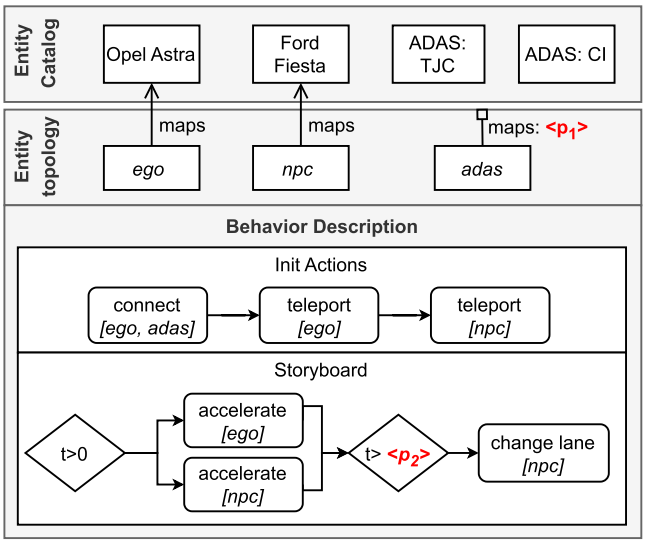


FIGURE 4. Example of an automotive abstract scenario. Parameters p_1 and p_2 are left undefined, so that the scenario can be concretized with 1) TJC or CI ADAS or 2) different trigger times for "change lane" action.

whereas the Story section specifies a lane-change maneuver with respect to the vehicle entities. Here, the choice of ADAS component variant (p_1: TJC or CI model) and the timing of the lane change maneuver (p_2) are left as undefined parameters. In practice, multiple abstract scenarios are defined to capture the full range of operational situations specified by the ODD.

2) LOGICAL SCENARIO

The logical scenario transforms an abstract scenario into a well-defined scenario space by assigning value ranges, probability distributions, and inter-parameter constraints to its

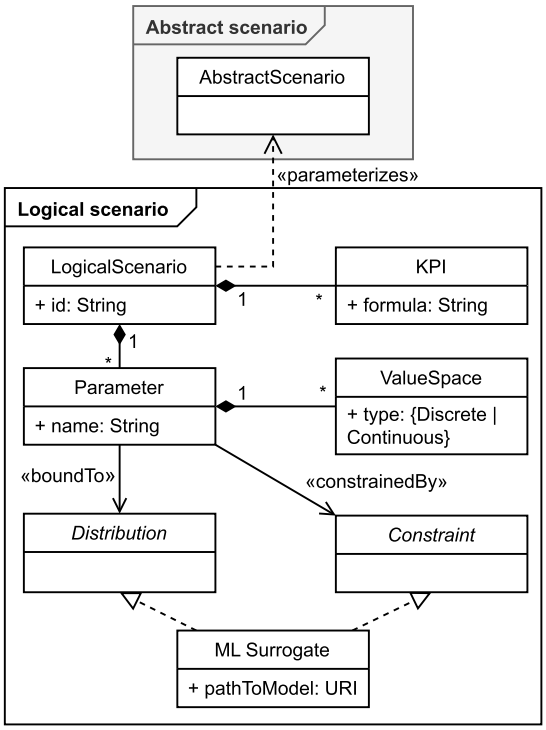


FIGURE 5. Meta-model for logical scenario.

undefined parameters. Following the meta-model in Figure 5, it is specified as an XML document that formally describes the parameter space.

Each parameter is defined through one or more `ValueSpace` elements (discrete or continuous ranges), bounded by `Distribution` specifications (e.g., Gaussian, uniform), and restricted through `Constraint` definitions (e.g., linear equalities, logical conditions). Beyond standard statistical formulations, the formalism supports closed-box machine learning models to capture complex parameter dependencies. Additionally, key performance indicators (KPIs) for evaluating simulation outcomes are specified at this stage, establishing the complete exploration problem: parameter space definition, evaluation metrics, and constraints.

This structured specification enables the logical scenario to serve as a dynamic framework for iterative refinement. As training or validation procedures identify critical regions or boundary conditions, constraints can be adjusted to guide subsequent sampling toward regions of interest.

3) CONCRETE SCENARIO

Finally, a sampler generates variants of the abstract scenario as *concrete* scenarios, each characterized by a unique combination of parameters sampled from the scenario space. For efficient coverage of the scenario space, the sampling process is integrated within an iterative scenario space exploration procedure wherein intelligent algorithms can actively search the scenario space for particular outcomes (detailed in Section V-B).

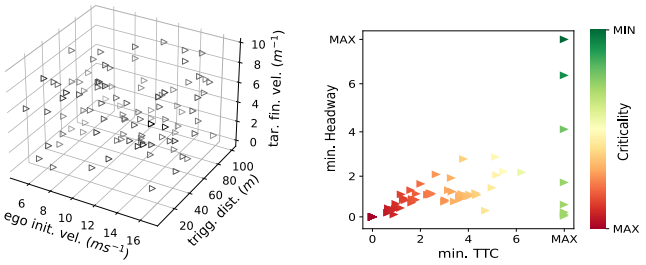


FIGURE 6. Visual analysis of the scenario space and outcome space coverage: (left) sampled concrete scenarios from scenario space, (right) evaluated DT-scenarios in outcome space.

C. SIMULATION OF DT-SCENARIOS

For simulation, concrete scenarios are instantiated as DT-scenarios which are executed within virtual testbeds. Herein, scenario entities are instantiated as their referenced DTs from the DT-library and scenario actions are translated to DT actions. The actions within the DT-scenario are structured via maneuver groups and event triggers to map the event flow of the simulation to the concrete scenario.

As a multitude of concrete scenarios is generated, the simulations take place on a compute cluster. Each simulation generates an event log containing time-series of properties already pre-specified in the concrete scenario.

D. SCENARIO EVALUATION

Given a concrete scenario with a parameter vector $\underline{p}$ and its corresponding DT-scenario simulation $\mathbf{M}$, a set of simulation outcomes $\underline{y}$ is calculated as

$$\underline{y} = \phi(\mathbf{M}(\underline{p})), \quad \underline{p} \in \mathbb{P}, \tag{1}$$

where $\mathbb{P}$ is the complete scenario space and ϕ is a key performance indicator (KPI) operating on the event logs of $\mathbf{M}$.

The sampled parameters $\{\underline{p}(0), \ldots, \underline{p}(j)\}$ and the outcomes $\{\underline{y}(0), \ldots, \underline{y}(j)\}$ provide an explicit measure of the scope of training data or validation generated by the simulations. An automotive example is illustrated in Figure 6. The abstract scenario "cut in from left and brake in front" is parameterized with respect to the initial velocity of the ego vehicle, trigger distance at which the target vehicle cuts in and the final velocity of target after braking. The sampled concrete scenarios are simulated and evaluated with respect to the minimum recorded time-to-collision and time-headway between the two vehicles in order to evaluate the criticality of each scenario. The figure shows that a uniform coverage of the scenario space results in a sufficient coverage of "critical" scenarios, whereas "non-critical" scenarios are only marginally covered.

V. STRUCTURED TRAINING AND VALIDATION OF AI-BASED SYSTEMS

Structured training and validation exploits the adaptability of DT-scenarios to different levels of abstraction. The possibility of extensive parameterization of DT-scenarios allows systematic scaling across the complexity of the SoI, complexity of

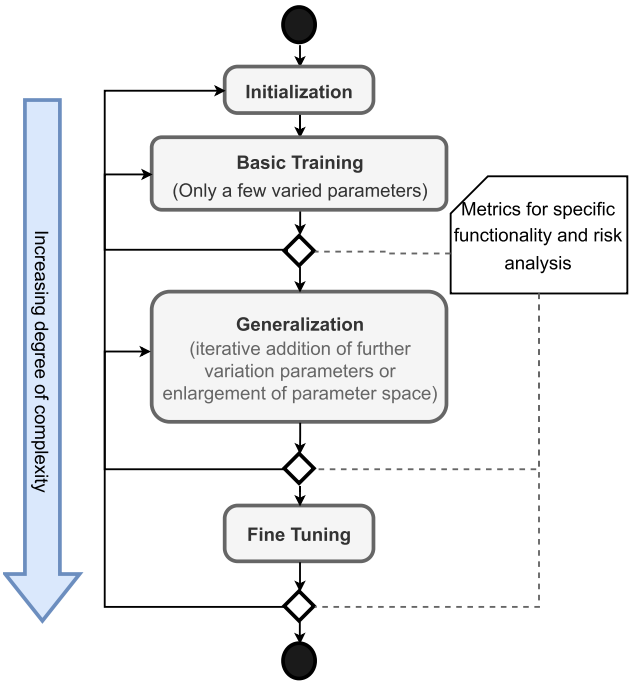


FIGURE 7. AI training process.

the operational situation and the *difficulty* of the operation. The following sections highlight the techniques applied for training and validation.

A. SUCCESSIVE COMPLEXITY INCREASE FOR STRUCTURED TRAINING

Particularly suited for the use-case presented in Section VI-A, we employ the principle of increasing complexity in which the target model progressively learns more challenging behaviors. The DT-scenarios simultaneously evaluate the learned behavior and generate training data for the next, more complex iteration.

The general process is described in Figure 7, and is applicable for both classic supervised learning applications and reinforcement learning. During the basic training, a small number of scenario-parameters are varied in a targeted fashion with the aim to learn the central characteristics of the feature space. In the generalization step, the complexity is increased iteratively by extending the parameter space of the scenario and therewith the area of application to be covered by the AI. Between the respective development steps, transfer learning is used to generalize the skills that have already been learned. Moreover, metrics for evaluating functionality and analyzing risk are used to evaluate whether current results are sufficient or whether individual sections need to be repeated. Finally, the AI can be further optimized using classic fine-tuning methods.

During each development step, an evaluation dataset is generated in addition to training data, which is used by the evaluation metrics to measure the performance of the currently trained model. Depending on the metric, both general behaviors and specific corner cases can be examined in detail. If a criterion is not met, further corresponding training data can be generated until satisfactory results are achieved.

B. SCENARIO SPACE EXPLORATION FOR COMPREHENSIVE VALIDATION

The primary goal of validation is to sufficiently cover the operational scenario space to ensure reliable performance of the SoI. However, exhaustive coverage is computationally infeasible, so a surrogate model is trained instead. The surrogate provides a statistical estimate of the SoI's performance as well as guides iterative exploration of the scenario space. An additional advantage of using a surrogate model is that it enables the entire search and validation process to remain automated and formally structured.

Formally, the KPI $\phi : M(\mathbb{P}) \rightarrow \mathbb{Y}$ from Eq. 1 evaluates the simulation outcome for a concrete scenario $\underline{p} \in \mathbb{P}$. The surrogate model $\Theta : \mathbb{P} \rightarrow \mathbb{Y}$ is conditioned on the evaluations $Y = \{\underline{y}_0, \ldots, \underline{y}_j\}$ of the sample-set $P = \{\underline{p}_0, \ldots, \underline{p}_j\}$ from the scenario space. The surrogate estimates are then used to suggest new sub-regions $\tilde{\mathbb{P}}_{i+1}$ of the scenario space that satisfy a search criteria ψ,

$$\tilde{\mathbb{P}}_{i+1} := \{\tilde{\mathbb{P}}_{i+1} \in \mathbb{P} \mid \psi\left(\theta(\tilde{\mathbb{P}}_{i+1})\right) \text{ is True}\}, \qquad (2)$$

where the search criteria ψ is a measure of the likelihood of finding *features* of the scenario space with respect to the evaluation metric. These features can be regions of best/worse performance, or boudary regions between different behavior types. The new sub-space $\tilde{\mathbb{P}}_{i+1}$ is added as a formal inter-parameter constraint in the logical scenario. Therefore, concrete scenarios from this sub-space are systematically generated through the logical scenario in the next iteration, while respecting any initial constraints specified. Equation 2 does not prescribe any particular model for the surrogate, rather this is chosen tailored to the problem at hand. For instance, Section VI-A demonstrates the usage of two surrogate models for guided scenario space exploration.

VI. CASE STUDIES

Two complementary case studies regarding training and validation of respectively aerospace and automotive systems are presented. Both case-studies have been carried out as industrial collaborations with specific requirements and goals. The first deals with the generation of device- and domain-specific training data for aerospace rendezvous and docking maneuvers. The second demonstrates function-level validation of a closed-box autonomous driving assistance system by learning surrogate models over a parameterized scenario space, given precisely defined operational design domain and safety metrics.

A. TRAINING RENDEZVOUS AND DOCKING MODEL

Automated rendezvous and docking (RvD) operations are playing an increasingly pivotal role in spaceflight missions. Beyond established resupply missions to the International

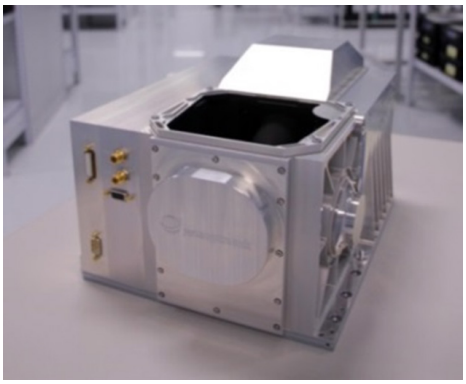


FIGURE 8. The RVS®3000 LiDAR scanner engineering model.

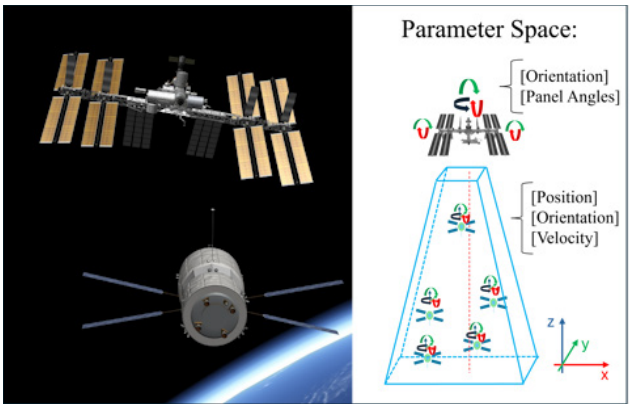


FIGURE 9. The rendezvous and docking use-case. Left figure shows the procedure, right figure illustrates the parameter space constructed by the logical scenario.

Space Station (ISS), new operational paradigms are emerging. The successful deployment of the Mission Extension Vehicle [33] has demonstrated the feasibility of on-orbit servicing for satellites and active debris removal. Additionally, the Artemis program plans to construct the Gateway in lunar orbit to serve as a logistics hub and steppingstone to Mars. Assembling and sustaining the resupply of such infrastructure will require a multitude of precise RvD maneuvers.

To ensure safety and redundancy, these missions rely on diverse sensor suites, including stereo camera systems and LiDARs. This case study focuses on a specific industrial application of the RVS®3000-3D LiDAR system, which was developed by Jena-Optronik GmbH (see Figure 8). The RVS®3000-3D is a time-of-flight scanning LiDAR with extensive flight heritage in missions involving the ATV, HTV, and Cygnus spacecraft [34]. Designed to detect and measure non-cooperative targets, the RVS®3000-3D uses a flexible scan pattern that can be continuously adapted enables a high-density point measurement and an operational range from one kilometer down to the final docking phase [33].

The sensor's core localization logic relies on an Iterative Closest Point (ICP) algorithm. This algorithm estimates the relative pose by matching a sampled point cloud from the sensor against a reference model generated from the target's structural data. However, a significant constraint of the ICP methodology is its iterative nature, which requires an accurate a priori estimate of the initial pose for correct convergence. Traditionally, this initialization has relied on external mission specifications or separate sensors. To mitigate this dependence and increase autonomy, Jena-Optronik has investigated machine learning approaches for initial pose estimation, first presented at i-SAIRAS 2018 [35].

1) PROBLEM DESCRIPTION

The pose estimation algorithm by Jena-Optronik utilizes a Fully Connected Neural Network (FCNN). The process involves sampling a subset of points from the 3D point cloud to form a "ScanPolygon". The FCNN then takes this ScanPolygon as input and is tasked with predicting the "MatchPolygon" the corresponding position of the vertices within the unrotated and untranslated reference frame of the target model. This prediction facilitates the computation of an initial transformation matrix, which seeds the ICP algorithm. If the subsequent ICP refinement yields an error below the defined threshold, the pose is accepted. Otherwise, initialization attempts are repeated with another randomly sampled "ScanPolygon".

A major challenge in developing this AI-based solution for space applications is the scarcity of accessible training data. It is rarely feasible to generate ground-truth data during actual missions, and real operating data covering the necessary variability is unavailable during the development phase. To complement the scarcity of data, our framework was used to create a training data generation pipeline was developed in collaboration with Jena Optronik and tailored to the operational constraints of the RVS®3000-3D sensor. By utilizing the DTs of the ISS and the RVS®3000-3D within the process defined, we generate large quantities of realistic, labeled point clouds to train the FCNN through successive levels of scenario complexity. This successive complexity increase also allows the study of the effects of covering different sections and conditions of the operational domain, lessons useful for further development of the product. For simulation, the DT of the ISS utilizes CAD models to render realistic geometries for scanning, whereas the DT of RVS®3000-3D is calibrated to match the scan pattern of its real counterpart. The development and calibration of the RVS®3000-3D DT has been detailed in related works [36].

2) INITIALIZATION OF THE TRAINING PROCESS

The first step of training data generation involves the specification of the neural network, the scenario design formalized as abstract and logical scenario, and evaluation metrics for measuring the quality of trained model.

The abstract scenario is illustrated in Figure 9 and describes a constantly moving chaser towards a static target. The chaser uses other sensors and mission specifications to navigate to an "approach corridor", a 400m long pyramid-shaped area delineating the operational domain of the RVS®3000-3D scanner. The scanner estimates the satellite pose to enable guidance towards the vicinity of the

TABLE 1. Parameter space description RvD.

Parameter	Value Range	Distribution	Training Phase
issRot_x	$[-0.17 : 0.17]$ rad	uniform	1
issRot_y	$[-0.17 : 0.17]$ rad	uniform	1
issRot_z	$[-0.17 : 0.17]$ rad	uniform	1
panelsRot_x	$[-3.00 : 3.00]$ rad	uniform	3
panelsRot_y	$[-3.00 : 3.00]$ rad	uniform	3
chaserPos_x	$[-70.0 : 70.0]\ m$	uniform	1
chaserPos_y	$[-70.0 : 70.0]\ m$	uniform	1
chaserPos_z	$[0.00 : 400.0]\ m$	uniform	1
chaserRot_x	$[-0.34 : 0.34]$ rad	uniform	2
chaserRot_y	$[-0.34 : 0.34]$ rad	uniform	2
chaserRot_z	$[-0.68 : 0.68]$ rad	uniform	2
chaserVel_x	$[0.00 : 0.5]\ m/s$	uniform	1
chaserVel_y	$[0.00 : 0.5]\ m/s$	uniform	1
chaserVel_z	$[0.00 : 0.5]\ m/s$	uniform	1
$\text{chaserPos}_i + 17.63 * \text{chaserPos}_z \geq -3$			$\forall i \in [x, y]$
$\text{chaserPos}_i - 17.63 * \text{chaserPos}_z \leq 3$			$\forall i \in [x, y]$

docking port. Final, high-precision docking maneuvers are then executed by dedicated close-range control systems.

Within the logical scenario, the parameter ranges are constrained to ensure the chaser remains within the truncated pyramid-shaped approach corridor and moves at a low, constant speed. Table 1 illustrates the variation parameters defined by the abstract and logical scenario. The ISS is allowed to rotate up to 10 degrees along each of three rotational axes. Its panels can also be rotated up to 270 degrees in the *xy*-plane. The chaser's translational parameters, together with linear constraints, ensure it remains within the approach corridor. The chaser is also allowed initial rotation of up to 20 degrees facing the target, and can move at speeds up to 0.5 meters per second.

A concrete scenario derived from this can then be executed in a virtual test-bed. Here, the digital twins are positioned according to the specifications, the laser scanner runs its scan pattern and the point cloud is saved. The generation and subdivision of training, test and validation data is then fully automated based on the simulation data. Each training or validation datum contains a randomly sampled partial point cloud, the so-called "ScanPolygon" from the point cloud. The ground truth in contains the positions of the ISS and the scanner.

The mean squared error (MSE) between the predicted and ground-truth points on the reference object is used as a quality metric during training and evaluation of the neural network. MSE does not directly represent the accuracy of the final pose estimate; rather, lower MSE values indicate a higher likelihood of successful convergence of the subsequent ICP refinement step.

3) BASIC TRAINING

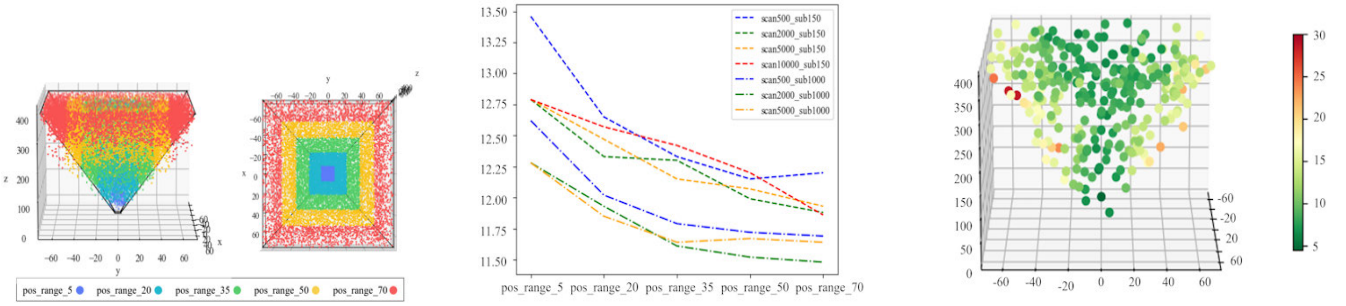
During the first stage of the training procedure (see Figure 7 for reference), the ISS is rotated within defined parameter ranges, the chaser position is varied within the approach corridor and its orientation is kept fixed aligned perpendicular to the ISS. The basic training is carried out in multiple training steps. Following the principle of iterative complexity increase, the approach corridor is expanded piece by piece through each step until the entire region is covered (Figure 10a). Each training step further contains multiple variants of training data sets. These sub-variants differ in the number of total scans (each scan corresponding to a single concrete scenario) and the number of partial point clouds sampled per scan. Figure 10b illustrates the comparison of the different variants through successive training steps. A higher number of partial point clouds from a scan leads to smaller error values. The number of total scans, on the other hand, brings no further improvement after 2000 scans per training step. Figure 10c shows that the accuracy of the prediction decreases the closer the chaser gets to the edge of the approach area. This can be attributed to the fact that recorded point clouds no longer completely capture the ISS at the edge of the corridor.

TABLE 2. Mean squared error values (m^2) from neural nets trained and tested on specific datasets.

	Test Datasets			
Neural Nets	Chaser vertical	Chaser rotated	mixed	Panels rotated
Chaser vert.	11.69	18.56	17.44	26.49
Chaser rot.	13.34	13.59	18.18	22.87
mixed	13.06	13.34	17.96	23.39
panels rot.	14.02	14.65	18.27	19.13
baseline	15.07	15.38	19.81	22.63

4) GENERALIZATION STEP

Building upon the basic training, further variation parameters are added and new training data is generated. In this phase, the chaser orientation is not necessarily fixed perpendicular to the target but can be rotated around its center of mass. Similarly, the solar panels of the ISS can also be rotated. A total of four variants of training and validation datasets are generated. Correspondingly, four neural networks are trained and subsequently validated on the validation data corresponding to their training dataset, as well as on other datasets to assess their generalization across different scenario distributions. Table 2 shows the average MSE values for each validation run. The attribute "vertical" indicates scenario-set from scenarios where the chaser orientation was kept fixed perpendicular to the target. "Rotated" scenario-set on the other hand indicates scenarios with varied chaser rotations within predefined ranges. "Mixed" scenario-set contains a mixture of the first two variants. The fourth variant includes scenarios where the panels of the ISS were also rotated. The baseline row refers to the performance of the neural network trained on existing datasets with the LiDAR manufacturer. Generally, each variant outperforms the manufacturer baseline. The exception is on datasets with rotated panels, which significantly modify



(a) Enlargement of the approach corridor through each training step: First, the chaser is positioned along the Z axis with deviations of +-5 m in the X and Y directions. The positioning range is then extended further in a total of 5 iterations.

(b) Error value through iteration steps depending on the number of total scans and extracted partial point clouds per scan.

(c) Color-coded errors of the network predictions while scanning from different positions of the approach corridor.

FIGURE 10. Basic training data generation and interim results.

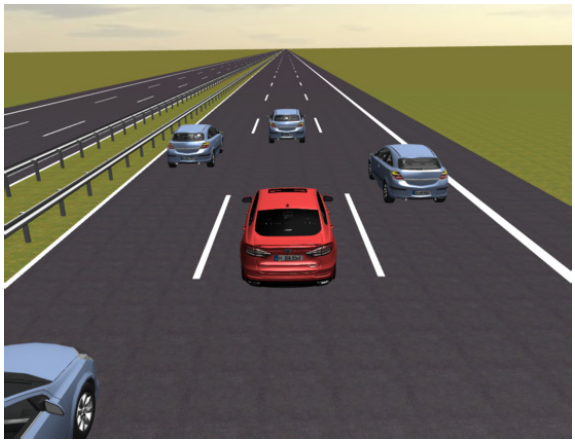


FIGURE 11. Simulation setup for validating traffic jam chauffeur.

the geometry to the target. Among the simulated variants, the neural networks perform best on their specific validation datasets, the mixed trained network also delivers results with similar error values on the respective base datasets. The fourth variant with randomly rotated panels shows a significant rise in scenario difficulty, as the first three neural networks perform poorly in this validation dataset. The neural network of the fourth variant delivers significantly lower MSE with its own validation dataset, but performs somewhat poorly on other validation datasets. The greatest possible flexibility with regard to the parameterization of the scenario is therefore directly associated with a loss of accuracy in specific areas.

5) LIMITATIONS AND SCOPE

The aerospace case-study opens up several directions of further research and also exposes some limitations of the current approach. First, we observed a plateau of performance gain beyond approximately 20,000 training samples after which the MSE shows minimal further improvement. This indicates that increasing the volume of training data alone does not further refine the network predictions for the given architecture. Second, partial point clouds used for training and validation are constructed by randomly sampling points from each scan. While increasing the number of sampled points per scan improves performance, as illustrated in Figure 10b, it also increases computational demands on the onboard processor of the RVS®3000-3D. This suggests that feature-aware point selection strategies may offer a better trade-off between accuracy and computational cost. Finally, the observed training behavior is specific to the employed network architecture, and alternative architectures may exhibit different data efficiency.

B. ADAS VALIDATION

This use-case presents the validation of ‘‘Traffic Jam Chauffeur’’ (TJC), an AI-based function of an autonomous driving assistance system (ADAS) that controls the vehicle in highway traffic jams. As part of the research collaboration, the TJC is delivered from a partner in the automotive industry as a closed-box module. The goal of this study is to provide a preliminary, simulation-based characterization of TJC behavior across its operational design domain (ODD) into *safe*, *moderate* and *critical regions*. Beyond aggregate performance estimates, the study aims to identify boundary regions between acceptable and unacceptable behavior such as ‘‘near-miss’’ events. Such regions are of particular interest for targeted testing and risk mitigation, and inefficiently explored by uniform random sampling.

The validation approach adopted in this case study is consistent with scenario-based workflows advocated by ISO 21448 (SOTIF) and the PEGASUS project, which emphasize ODD definition, scenario-based testing, and KPI-driven evaluation of intended functionality. While these frameworks do not prescribe concrete methods for scenario space exploration, the presented case study operationalizes these steps

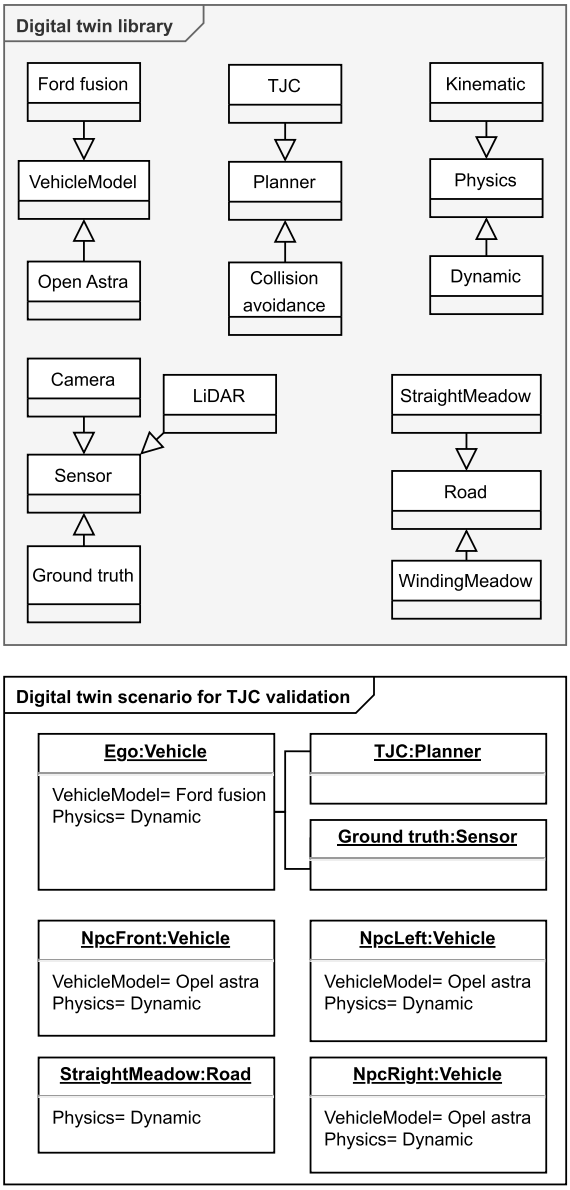


FIGURE 12. Digital twin library and scenario instantiation for Traffic Jam Chauffeur (TJC) validation. The library defines reusable system components (e.g., vehicle models, physics, sensors, planners), while the digital twin scenario is generated by selecting and composing specific component variants. This separation enables reuse of abstract scenarios across different system configurations and validation stages.

through iterative, surrogate-guided simulation to support targeted exploration of safety-relevant regions of the scenario space without requiring access to internal system states.

1) INTEGRATION IN DT-LIBRARY

The TJC is delivered as a closed-box module and integrated within the simulation model of the ego vehicle. This is realized via the DT concept explained in Sec III, and illustrated in Figure 12. The DT-library for this case study contains the models of different aspects of traffic simulation, for instance vehicle models developed from different manufacturers, physics models including kinematic or dynamic-based, sensor models including camera, LiDAR and ground truth, planners including a simplistic collision avoidance and the delivered TJC, and road models including straight or curved highways.

From the DT-library, the DT-scenario for TJC validation is created by instantiating particular variants of each of the DTs. Since the objective of this study is function-level validation of the planning component, the ego vehicle is equipped with a ground-truth sensor that provides logical scene information directly to the Traffic Jam Chauffeur (TJC) planner. This configuration isolates planning behavior from perception effects and enables focused analysis of decision-making within the defined operational design domain. The ego vehicle is assigned a Ford Fusion vehicle model with dynamic physics, while surrounding traffic participants are modeled as dynamic vehicles without sensing or planning capabilities, as their behavior is fully defined by scenario actions.

Importantly, DT-scenarios are not modelled, rather automatically generated by specifying the appropriate entity-to-DT mappings and inter-connections in the abstract scenario, as was detailed in Section IV.

2) SCENARIO SPECIFICATION

The operational scenario types spanning the operational design domain of TJC are specified as abstract scenarios with OpenSCENARIO formalism. An abstract scenario defines at-least three non-ego vehicles: initialized on the left, right and front (termed `npc.left`, `npc.right`, `npc.front` respectively) of the ego to simulate a traffic jam situation. Figure 13 illustrates the actions performed by `npc` vehicles, which serves as building block for describing abstract scenarios. A detailed break-down of the specified abstract scenarios, together with their actions and undefined parameters is provided in Appendix A (Table 5).

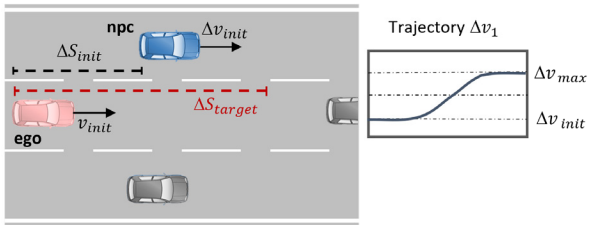
The parameter space $\mathbb{P}$ of each abstract scenario constitutes of its parameters $\underline{P}$, and is specified in the logical scenario. Each parameter is herein assigned min. and max. ranges, and G specifies inter-parameter relation constraints necessary for the desired execution of the scenario.

$$\mathbb{P} := \begin{cases} p_{in} \leq p \leq p_{max} & \forall p \in \underline{P}, \\ G(\underline{P}) = True \end{cases} \tag{3}$$

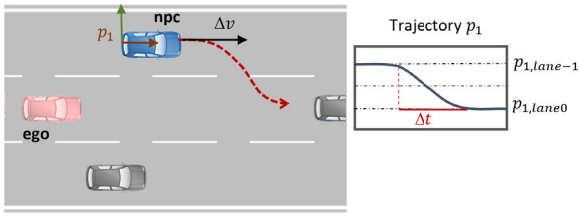
The ranges and constraints in Equation 3 are initially derived through intuition. They ensure that uninteresting phenomena (e.g. collision between non-ego vehicles) are avoided, and aim to cover both known *safe* and *critical* regions of operation.

3) ITERATIVE SCENARIO SPACE EXPLORATION

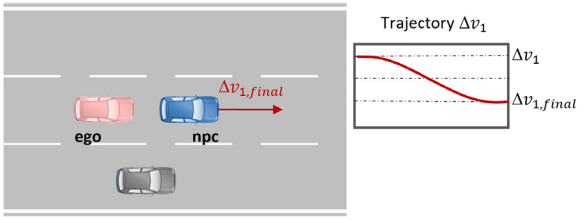
The objective of iterative scenario space exploration is to employ surrogate-guided sampling to concentrate simulations in regions of interest. Surrogate models are first trained on an initial batch of randomly sampled scenarios, then used to identify regions where *critical* or transition behavior is predicted, and these regions guide the generation of subsequent scenario batches. We demonstrate this via two surrogate models: the support vector machine (SVM) and Gaussian



(a) Longitudinal distance action: npc accelerates constrained by max acceleration and velocity until it reaches target distance to relative vehicle.



(b) Lane change action: npc performs lane change maneuver parameterized by the duration of the maneuver.



(c) Speed action: npc speeds up or down constrained by the maximum acceleration or deceleration.

FIGURE 13. Examples of atomic OpenSCENARIO actions used to build up scenarios within the operational design domain of traffic jam chauffeur.

processes (GP) [37], [38]. The purpose of employing multiple models is not to identify an optimal surrogate model, but to demonstrate that the framework itself is model-agnostic and that different models can be used to provide comparable gains over random exploration. As the same general methodology is employed for all operational scenarios in Table 5, therefore we limit the presentation to two use-cases, a ''cut in and brake in front'' scenario with 3-dimensional parameter space and a more challenging ''cut out with obstacle in front'' scenario with a 6-dimensional parameter space.

a: Cut in and brake in front

This scenario features a npc vehicle speeding up to reach a certain distance ahead of the ego, cutting in front and slowing down. Three parameters are varied: the initial speed of ego vehicle, trigger distance at which the npc vehicle cuts in, and final speed of the npc vehicle to which it slows down. In the first step, the first batch of concrete scenarios is uniformly sampled from the respective scenario space defined in Equation 3 using rejection sampling methods, and simulated on a high performance compute cluster. For surrogate-guided exploration, the metric ''time-to-collision'' (TTC) is used, which describes the duration in which two vehicles are projected to collide while keeping their current trajectories.

Three categories are defined based on minimum recorded TTC. *Critical* scenarios have a TTC of 0, indicating a collision occurred. *Moderate* scenarios have a TTC between 0 and 5 seconds. *Safe* scenarios have a TTC greater than 5 seconds. The threshold of 5 second was chosen to distinguish potentially hazardous scenarios necessitating evasive action or warning activation, based on traffic surveys [39]. Figure 14a illustrates the concrete scenario evaluations for the operational scenario ''cut in and brake in front'' within the three-dimensional scenario space.

In the second step, the sampled batch of concrete scenarios and their respective outcomes are used as training samples for the SVM and GP surrogates. SVM is trained with the afore-mentioned outcome categories, whereas the GP is trained directly on the numerical TTC values. The trained surrogates can thereafter predict the outcome over the complete scenario space. Figure 14b illustrates this for SVM.

The surrogates are then used for further exploration of the scenario space in a targeted fashion. This is formally done by specifying new conditions ϕ in the logical scenario with respect to the respective surrogate θ, thereby restricting the original feature space to those satisfying the conditions:

$$\tilde{\mathbb{P}}_1 := \begin{cases} p_{in} \leq p \leq p_{max}, & \forall p \in \underline{P}, \\ G(\underline{P}) = \text{True}, \\ \phi\left(\Theta(\underline{P})\right) = \text{True}. \end{cases} \tag{4}$$

where ϕ specifies the boundary between *critical* and *moderate* scenarios, i.e. scenarios with ''near misses''. For SVM, ϕ is a constraint on the prediction probabilities[1] of the model,

$$\phi(\Theta_{\text{SVM}}) = \mathbf{1}\left\{ \begin{matrix} p_2^{\Theta_{\text{SVM}}} < \alpha, \\ \beta \leq p_1^{\Theta_{\text{SVM}}} \leq \gamma \end{matrix} \right\}, \tag{5}$$

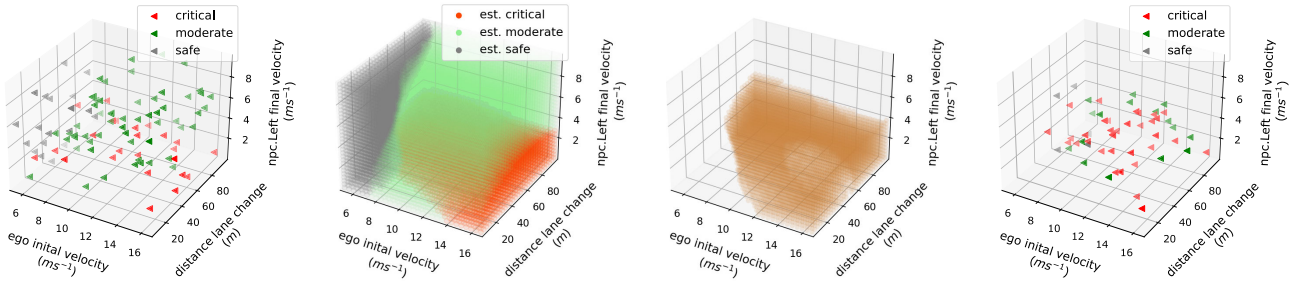
where p_2^{Θ} is the probability of *safe* event and p_1^{Θ} is the probability of a *moderate* event. Parameters α,β and γ are tuned such that the model is reasonably certain that *safe* outcome shall not occur, but remaining uncertain between *critical* and *moderate* outcomes. The resulting boundary is illustrated in Figure 14c. For GP, ϕ is defined over the TTC predictions of the model,

$$\phi(\Theta_{\text{GP}}) = \mathbf{1}\left\{0 < \text{TTC}^{\Theta_{\text{GP}}} \leq \tau\right\}, \tag{6}$$

where τ is chosen to tune the predicted severity of outcomes. Note that $TTC = 0$ is left out of the boundary to exclude direct collisions and encourage ''near miss'' type events.

The surrogate-conditioned parameter space in Equation 5 is sampled to yield concrete scenarios. A summary of evaluations is presented in Table 3, which compares random sampling and surrogate-guided sampling with respect to

[1]Prediction probabilities from the `scikit-learn` SVM library correspond to the distance of samples from the high-dimensional decision boundary [40].



(a) Simulation outcomes of first batch of uniformly distributed concrete scenarios

(b) SVM-based estimation of outcomes, conditioned on the first simulation batch

(c) Estimated boundary region between *critical* and *moderate* outcomes

(d) Simulation outcomes of second batch of concrete scenarios sampled from the boundary region

FIGURE 14. Different validation steps of the operational scenario "cut in and brake in front".

TABLE 3. Scenario "Cut in and brake in front": comparison of surrogate-guided exploration with random sampling with respect to concentration of simulations in a near-miss region (TTC $\in (0, \tau]$).

Sampling strategy	% TTC $\in (0, \tau]$	% Moderate	% Critical	% Safe
Random	8	26	17	57
Support Vector Machine	48.8	71.7	22.8	5.5
Gaussian Process	47.2	66.4	31.2	2.4

their ability to concentrate simulation effort in the boundary regions of interest specified as TTC $\in (0, \tau]$. The parameter τ does not redefine the safety categories used for evaluation. Instead, it specifies a focus window within the moderate region to concentrate simulation effort on near-miss behaviors. In this study, $\tau = 3$ was selected as a representative near-miss window to demonstrate the ability of surrogate-guided sampling to concentrate simulations closer to critical behavior.

Table 3 shows around 5x improvement in sampling boundary scenarios using surrogate guided explorations. The percentage of *moderate* scenarios is expectedly high as well, since the boundary definition TTC $\in (0, \tau]$ lies within the definition of this category. Critical scenarios are sampled more frequently than the baseline. This is expected, as the boundary region lies close to TTC $= 0$, and surrogate is bound to make misclassifications in this area. On the other hand, the uninteresting *safe* scenarios are starkly under-sampled in surrogate exploration, eliminating unnecessary compute resource consumption.

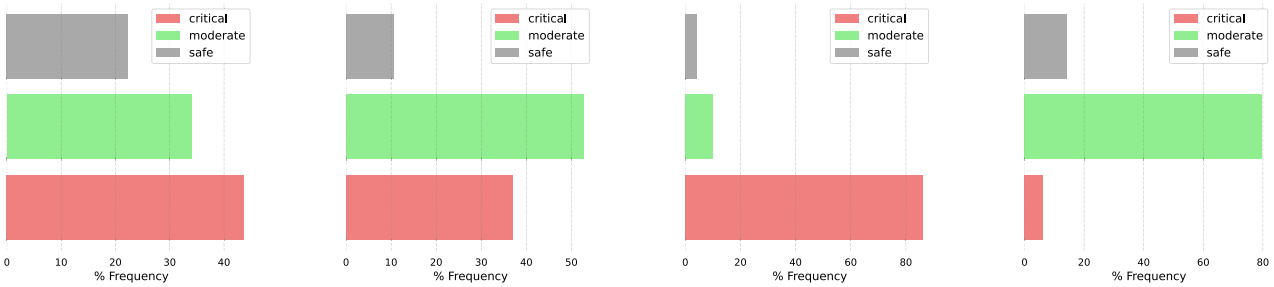
b: Cut out and obstacle in front

This scenario is more challenging than the first and demonstrates the effectiveness of surrogate-guided sampling in higher-dimensional spaces. The scenario involves two non-ego vehicles: `npc.FrontNear` driving ahead of the ego vehicle, and `npc.FrontFar` ahead of `npc.FrontNear` either stopped or driving at a low speed. When `npc.FrontNear` cuts out, the `npc.FrontFar` suddenly becomes visible to the ego vehicle and tests its reaction capabilities. The varied parameters include: speeds of the three vehicles, initial distances between vehicles, trigger distance at which `npc.FrontNear` cuts out, and the duration of the lane-change maneuver. Similar to the previous scenario, evaluations are categorized into *critical*, *moderate* and *safe* based on the time-to-collision metric.

TABLE 4. Effectiveness of SVM-guided sampling for the "Cut out with obstacle in front" scenario. The surrogate model enables targeted exploration by concentrating simulations in regions of interest based on specified conditions ϕ.

Sampling strategy	% Moderate	% Critical	% Safe
Random	24	54	22
SVM (ϕ = critical)	4.8	83.8	11.4
SVM (ϕ = moderate)	79.6	6.1	14.3

The evaluations for this scenario are summarized in Table 4 and illustrated in Figure 15. The first batch of randomly sampled concrete scenarios, illustrated in Figure 15a, showcases ample distributions of all three categories. An SVM model is trained on this batch, and is used to condition the future generation of three types of further batches. For generating critical scenarios, the condition $\phi = \textit{critical}$ is added to the logical scenario, resulting in about 1.5x more critical scenarios than random sampling (Figure 15c). Similarly, using the condition $\phi = \textit{moderate}$ results in around 3x more moderate scenarios compared to random sampling (Figure 15b). Similarly, using the boundary condition $\phi = \textit{boundary}$ through Equation 5 results in under-sampling of *safe* scenarios and higher percentages of both *critical* and *moderate* scenarios (Figure 15d).



(a) From uniformly distributed initial batch

(b) From estimated boundary regions between *critical* and *moderate* outcomes

(c) From regions with estimated only *critical* outcomes

(d) From regions with estimated only *moderate* outcomes

FIGURE 15. Simulation outcomes from different sub-regions of the feature space of the operational scenario "cut out with obstacle in front".

4) LIMITATIONS AND SCOPE

The TJC validation process demonstrates the effectiveness of data-driven surrogate models for function-level validation within a systematic scenario-based framework. At the same time, it exposes several limitations and directions for further investigation.

First, the presented validation focuses on the planning function using ground truth scene information, and does not account for perception-induced effects or end-to-end system behavior. A natural next step is to leverage the digital twin library to incorporate physically realistic sensor models and jointly study the interaction between perception and planning. The current results provide a useful starting point for such integrated testing, as scenarios associated with known planning failures can be deprioritized when analyzing perception-related effects.

Second, the number of parameters can quickly become infeasible for effective coverage or exploration in more complex studies. Here, additional strategies such as parameter sensitivity analysis need to be explored to manage dimensionality and to support adequate coverage of the scenario space.

VII. CONCLUSION AND DISCUSSION

This work presented a framework for the systematic training and validation of AI-based systems through an integration of digital twins and scenario engineering. The framework combines the scalability, interpretability, and traceability of classical scenario-based approaches with modular, high-fidelity simulation models that can be flexibly configured for both training and validation tasks. By introducing the concept of a digital twin scenario and a unified scenario definition formalism, the framework enables scenario-based experimentation across application domains, provided that domain-specific models can be expressed as digital twin scenario templates.

The two case studies demonstrate complementary roles of the framework across the system development life cycle. The aerospace case study shows how scenario-based simulation can be used to systematically generate application-specific training data and to study the influence of operating conditions on learning-based perception components. The automotive case study illustrates how data-driven surrogate models can be employed to explore and characterize system behavior over a parameterized scenario space and increase coverage of targeted behavior. Together, these studies show that scenario-based simulation can serve not only as a test execution mechanism, but as a structured means to reason about data generation, system behavior, and validation evidence.

A. LIMITATIONS AND FUTURE WORK

The presented work also highlights several open challenges that promise advances in fully realizing the potential of scenario-based digital twin validation in large-scale settings. First, the effectiveness of scenario exploration is strongly influenced by the choice of parameters and their interactions. For higher-dimensional problems, integrating techniques such as global sensitivity analyses can help identify dominant parameters and higher-order interactions [41], [42]. These interactions are not explicitly captured by the surrogate models considered here.

Second, the selection of abstract scenarios plays a central role in achieving coverage of a given operational design domain. While this problem has been studied in the automotive domain [9] and is treated as an external input in this work, extending such approaches across application domains remains a prerequisite for broadly applicable scenario-based validation.

Third, increased simulation fidelity typically comes at the cost of reduced computational efficiency. Although the proposed architecture allows configuring model fidelity through system design variation, systematically balancing fidelity, computational cost, and validation effectiveness remains an open research question. Addressing this challenge also motivates further work on scalable simulation infrastructures, including concurrent execution of high-fidelity simulations across distributed compute resources.

Finally, applying the framework to domains with accessible physical platforms, such as robotics, offers opportunities to complement simulation-based results with real-world

TABLE 5. Overview of the operational scenarios and their parameterization for validating the TJC module. All "relative" parameters are relative to ego unless otherwise specified.

Operational Scenarios	Major Actions	Abstract Parameters
Cut in	`npc.Left`: Reach longitudinal distance to ego → Lane change to ego lane	• Ego initial velocity • Trigger distance for lane change
Cut in and brake in front	`npc.Left`: Reach longitudinal distance to ego → Lane change to ego lane → Change speed (decelerate)	• Ego initial velocity • Trigger distance for lane change • Relative final velocity after brake
Cut out with obstacle in front (*`npc.frontNear` performs the cut out maneuver while `npc.frontFar` is a slow moving obstacle*)	`npc.frontNear`: Reach longitudinal distance to `npc.frontFar` → Lane change out of ego lane	• Ego initial velocity • `npc.frontNear` relative initial velocity • `npc.frontNear` relative initial distance • Trigger distance between `npc.frontFar` and `npc.frontNear` for lane change • Lane change time • `npc.frontFar` relative initial velocity • `npc.frontFar` relative initial distance to `npc.frontNear`
Crossing from left to right	`npc.Left`: Reach longitudinal distance to ego → Lane change across ego lane	• Ego initial velocity • Trigger distance for lane change • Lane change time • Relative velocity during lane change
Resolving traffic jam	*All npc vehicles:* Wait → Accelerate → Change speed (brake)	• Speed up acceleration • Speed down deceleration

experiments. Continued collaboration with industrial partners across domains will be essential for grounding these developments in practice and for informing the emergence of standardized validation and quality assurance workflows for AI-based systems.

LIST OF ABBREVIATIONS

Abbreviation	Definition
ADAS	Advanced Driver Assistance System.
AI	Artificial Intelligence.
CAD	Computer-Aided Design.
DT	Digital Twin.
FCNN	Fully Connected Neural Network.
GP	Gaussian Process.
ICP	Iterative Closest Point.
ISS	International Space Station.
KPI	Key Performance Indicator.
LiDAR	Light Detection and Ranging.
MSE	Mean Squared Error.
ODD	Operational Design Domain.
RvD	Rendezvous and Docking.
SoI	System of Interest.
SOTIF	Safety Of The Intended Functionality.
SVM	Support Vector Machine.
TJC	Traffic Jam Chauffeur.
TTC	Time-To-Collision.
XML	Extensible Markup Language.

APPENDIX

TJC VALIDATION

Table 5 provides a detailed overview of the operational scenarios and their parameterization for validating the Traffic Jam Chauffeur (TJC) module.

REFERENCES

[1] S. Ettinger et al., "Large scale interactive motion forecasting for autonomous driving : The waymo open motion dataset," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 9690–9699.

[2] H. Caesar et al., "NuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11618–11628.

[3] N. Wang, Y. Luo, T. Sato, K. Xu, and Q. A. Chen, "Does physical adversarial example really matter to autonomous driving? Towards system-level effect of adversarial object evasion attack," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2023, pp. 4389–4400.

[4] A. V. S. Neto, J. B. Camargo, J. R. Almeida, and P. S. Cugnasca, "Safety assurance of artificial intelligence-based systems: A systematic literature review on the state of the art and guidelines for future work," *IEEE Access*, vol. 10, pp. 130733–130770, 2022.

[5] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "VirtualWorlds as proxy for multi-object tracking analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4340–4349.

[6] B. Hurl, K. Czarnecki, and S. Waslander, "Precise synthetic image and LiDAR (PreSIL) dataset for autonomous vehicle perception," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2019, pp. 2522–2529.

[7] E. Alberti, A. Tavera, C. Masone, and B. Caputo, "IDDA: A large-scale multi-domain dataset for autonomous driving," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5526–5533, Oct. 2020.

[8] T. Sun et al., "SHIFT: A synthetic driving dataset for continuous multi-task domain adaptation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 21339–21350.

[9] F. Hauer, T. Schmidt, B. Holzmüller, and A. Pretschner, "Did we test all scenarios for automated and autonomous driving systems?" in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 1433–1438.

[10] C. Amersbach and H. Winner, "Defining required and feasible test coverage for scenario-based validation of highly automated vehicles," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 425–430.

[11] A. Eidehall and L. Petersson, "Statistical threat assessment for general road scenes using Monte Carlo sampling," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 1, pp. 137–147, Mar. 2008.

[12] M. O'Kelly, A. Sinha, H. Namkoong, J. C. Duchi, and R. Tedrake, "Scalable end-to-end autonomous vehicle testing via rare-event simulation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018.

[13] A. Corso, R. Moss, M. Koren, R. Lee, and M. Kochenderfer, "A survey of algorithms for black-box safety validation of cyber-physical systems," *J. Artif. Intell. Res.*, vol. 72, pp. 377–428, Oct. 2021.

[14] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski, "Simulation-based adversarial test generation for autonomous vehicles with machine learning components," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 1555–1562.

[15] T. Dreossi et al., "VerifAI: A toolkit for the formal design and analysis of artificial intelligence-based systems," in *Proc. Int. Conf. Comput. Aided Verification*, 2019, pp. 432–442.

[16] C. Watson, R. Alur, D. Gopinath, R. Mangal, and C. S. Păsăreanu, "Scenario-based compositional verification of autonomous systems with neural perception," in *Proc. Int. Symp. AI Verification*, 2025, pp. 3–28.

[17] International Organization for Standardization. (2019). *ISO/PAS 21448:2019-Road Vehicles-Safety of the Intended Functionality*. [Online]. Available: https://www.iso.org/standard/70939.html

[18] H. Winner, K. Lemmer, T. Form, and J. Mazzega, "PEGASUS—First steps for the safe introduction of automated driving," in *Road Vehicle Automation*. Cham, Switzerland: Springer, 2019, pp. 185–195.

[19] X. Zhang et al., "Finding critical scenarios for automated driving systems: A systematic mapping study," *IEEE Trans. Softw. Eng.*, vol. 49, no. 3, pp. 991–1026, Mar. 2023.

[20] I. Graessler, M. Ebel, and J. Pottebaum, "Model-based planning of test cases and test scenarios to support engineering of cyber-physical systems," in *Proc. IEEE Int. Symp. Syst. Eng. (ISSE)*, Oct. 2024, pp. 1–8.

[21] J. Cederbladh, A. Cicchetti, and J. Suryadevara, "Early validation and verification of system behaviour in model-based systems engineering: A systematic literature review," *ACM Trans. Softw. Eng. Methodology*, vol. 33, no. 3, pp. 1–67, Mar. 2024.

[22] (2023). *Asam Openscenario*. [Online]. Available: https://www.asam.net/standards/detail/openscenario/

[23] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Scenic: A language for scenario specification and scene generation," in *Proc. 40th ACM SIGPLAN Conf. Program. Lang. Design Implement.*, 2019, pp. 63–78.

[24] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, vol. 78, 2017, pp. 1–16.

[25] O. Michel, "Cyberbotics Ltd. Webots: Professional mobile robot simulation," *Int. J. Adv. Robotic Syst.*, vol. 1, no. 1, p. 5, Mar. 2004.

[26] (Jul. 2022). *Association for Standardisation of Automation and Measuring Systems (ASAM) E.V. Asam Openscenario*. [Online]. Available: https://www.asam.net/static_downloads/public/asam-openscenario/2.0.0/welcome.html

[27] M. Wild, J. S. Becker, C. Schneiders, and E. Möhlmann, "A scenario-based simulation framework for testing of highly automated railway systems," in *Proc. 11th Int. Conf. Vehicle Technol. Intell. Transp. Syst. (VEHITS)*. Porto, Portugal: SciTePress, Apr. 2025, pp. 88–99. [Online]. Available: https://doi.org/10.5220/0013286600003941

[28] J. Rossmann, M. Schluse, C. Schlette, and R. Waspe, "Control by 3D simulation–A new robotics approach to control design in automation," in *Proc. Int. Conf. Intell. Robot. Appl.*, 2020, pp. 120–130.

[29] O. Maqbool and J. Roßmann, "Formal scenario-driven logical spaces for randomized synthetic data generation," in *Proc. 10th Int. Conf. Model-Driven Eng. Softw. Develop.*, 2022, pp. 203–210.

[30] O. Maqbool and J. Rossmann, "Virtual scenario testbeds for formal training and validation of machine-learning based systems," in *Proc. 37th Eur. Simul. Model. Conf.*, 2023, pp. 223–230.

[31] U. Dahmen, T. Osterloh, and J. Roßmann, "Structured validation of AI-based systems by virtual testing in simulated test scenarios," *Int. J. Speech Technol.*, vol. 53, no. 15, pp. 1–15, Aug. 2023.

[32] M. Schluse, *Experimentierbare Digitale Zwillinge: Konvergenz Von Simulation Und Realität*. Cham, Switzerland: Springer, 2024.

[33] N. T. Redd, "Bringing satellites back from the dead: Mission extension vehicles give defunct spacecraft a new lease on life," *IEEE Spectr.*, vol. 57, no. 8, pp. 6–7, Aug. 2020.

[34] Jena-Optronik and A. Feige. (Sep. 2013). *New Cargo Logistics Spacecraft 'Cygnus' Guided Safely To the International Space Station (ISS) With Technology From Jena*. [Online]. Available: https://www.jena-optronik.de/en/news/reader/cygnus-en.html

[35] C. Christoph, J. Both, and F. Kolb, "RVS3000–3D: LiDAR meets neural networks," in *Proc. Int. Symp. Artif. Intell., Robot. Autom. Space*, 2022, pp. 1–7.

[36] J. Thieling, "Chapter 8: Anwendung und evaluation der architektur," in *Architektur Zur Entwicklungsbegleitenden Modellierung Und Simulation Von Umfeldsensoren in Hybriden Testbeds*. Cham, Switzerland: Springer, 2023, ch. 8, pp. 161–212.

[37] B. Schölkopf and A. J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2002.

[38] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.

[39] K. Vogel, "A comparison of headway and time to collision as safety indicators," *Accident Anal. Prevention*, vol. 35, no. 3, pp. 427–433, May 2003.

[40] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[41] M. Owais and G. S. Moussa, "Global sensitivity analysis for studying hot-mix asphalt dynamic modulus parameters," *Construction Building Mater.*, vol. 413, Jan. 2024, Art. no. 134775.

[42] L. K. Idriss and M. Owais, "Global sensitivity analysis for seismic performance of shear wall with high-strength steel bars and recycled aggregate concrete," *Construct. Building Mater.*, vol. 411, Jan. 2024, Art. no. 134498.