Applied Intelligence (2023) 53:18910–18924
https://doi.org/10.1007/s10489-023-04475-x

Structured validation of AI-based systems by virtual testing in simulated test scenarios

Ulrich Dahmen[1] · **Tobias Osterloh**[1] · **Jürgen Roßmann**[1]

Accepted: 15 January 2023 / Published online: 14 February 2023

Abstract
The growing relevance of artificial intelligence (AI) for technical systems offers significant potential for the realization and operation of autonomous systems in complex and potentially unknown environments. However, unlike classical solution approaches, the functionality of an AI system cannot be verified analytically, which is why data-driven approaches such as scenario-based testing are used. With the increasing complexity of the required functionality of the AI-based system, the quantity, and quality of the data needed for development and validation also increase. To meet this demand, data generated synthetically using simulation is increasingly being used. Compared to the acquisition of real-world reference data, simulation offers the major advantage that it can be configured to test specific scenarios of interest. This paper presents an architecture for the systematic generation of virtual test scenarios to establish synthetically generated test data as an integral part of the development and validation process for AI systems. Key aspects of this architecture are the consistent use of digital twins as virtual 1-to-1 replicas and a simulation infrastructure that enables the generation of training and validation data for AI-based systems in appropriate quantity, quality, and time. In particular, this paper focuses on the application of the architecture in the context of two use cases from different application domains.

Keywords Digital twin · Modeling and simulation · Verification and validation · Virtual testbed

1 Introduction

Emerging from the rapid development of modern AI technologies, intelligence and autonomy are being used in mechatronic systems to an ever greater extent. This progress is supported by the ever-increasing potential of hardware performance and software technology. Especially through the use of AI, mechatronic systems are becoming increasingly complex and the need for comprehensive functional validation exists. The importance of functional validation quickly becomes apparent, considering that AI-based approaches commonly realize tasks like environment perception or are responsible for intelligent interaction with a potentially unknown operational environment. The resulting system behavior somewhat becomes opaque and emergent behavior can no longer be predicted with certainty.

In particular, AI-based approaches built on neural networks are characterized by their black-box nature. Consequently, well-known module-by-module verification approaches (e.g. from software engineering) are no longer applicable, stating a severe challenge for their functional validation. As a result, AI-based approaches are only established at the subsystem level (e.g. image processing) but not yet in system-level applications with harsh safety considerations. The potential of AI-based approaches is enormous, but the need for comprehensive quality assurance of AI-based systems persists. Consequently, more sophisticated processes for the end-to-end verification of AI-based systems are inevitably required.

Looking at current challenges in the development and validation of AI-based systems, some recurring points emerge. The required amount and needed variety of data is commonly acknowledged as a core challenge. Therefore,

Ulrich Dahmen and Tobias Osterloh are contributed equally to this work.

✉ Ulrich Dahmen
dahmen@mmi.rwth-aachen.de

✉ Tobias Osterloh
osterloh@mmi.rwth-aachen.de

Jürgen Roßmann
rossmann@mmi.rwth-aachen.de

[1] Institute for Man-Machine Interaction, RWTH Aachen University, Ahornstraße 55, 52074, Aachen, NRW, Germany

real-world tests of AI-based systems in their operational environment are typically envisaged (e.g. test drives for validation of AI-based driver assistant systems). They provide an accurate evaluation of the AI performance since the AI-based system is directly tested in its designated operational environment. Nevertheless, several drawbacks are not yet negligible: Real-world tests come at high costs (financially and timely) and might state risks for entities of the real environment (e.g. other road users). Additionally, it is not guaranteed that all relevant conditions or scenarios will occur during the real-world test drives (e.g. cyclist on the street while raining). Focusing on such scenario-based validation approaches, these incomplete data sets might result in malfunctioning AI-based systems.

Reflecting on these considerations and challenges, it is commonly accepted to accompany the validation process of AI-based systems with virtual test drives. Mandatory real-world tests are complemented by simulated tests that provide several advantages: Parameters and conditions can be purposefully specified, simulations are deterministic and reproducible, and simulation data can be generated cost-effectively and faster than in real-time. However, current simulation-centered approaches cannot usually formally describe scenarios and automatically generate scenario variations. In addition, existing approaches mostly focus on individual aspects of AI validation or are only suited for specific application domains. The basic concepts are only loosely coupled and have no conceptual base. A holistic framework for the structured cross-application development and validation process of an AI-based system does not yet exist. In this case, the validation approach presented in this paper meets this requirement. The goal of this research is to provide a conceptual framework for the structured and highly automated scenario-driven validation of AI-based systems, that helps to build trust in the correct functioning of the AI (see Fig. 1).

This contribution is an extended version of the following paper [1] and focuses on a detailed overview of the practice-oriented application of the presented architecture, highlighting the proposed iterative process for the holistic validation of AI-based systems. Consequently, the existing approaches and relevant aspects of the architecture are only briefly highlighted in Sections 2 and 3. The benefits and versatility of the proposed architecture are demonstrated in Section 4 by examining the validation of AI-based systems in an automotive application as well as in the often neglected application domain of aerospace engineering. The contribution concludes with a summary of the results in Section 5.

2 State of the art

The generation of synthetic data for the training and validation of AI-based systems requires several core technologies: formal descriptions of scenarios, a simulation framework, and standardized processes. The following section briefly compares and summarizes the relevant technologies.

2.1 Standardized description of scenarios

A scenario is a formalized description of a specific use case of a system and its operational environment. Currently, two major specification languages exist for the formal description of scenarios: the SCIENIC language

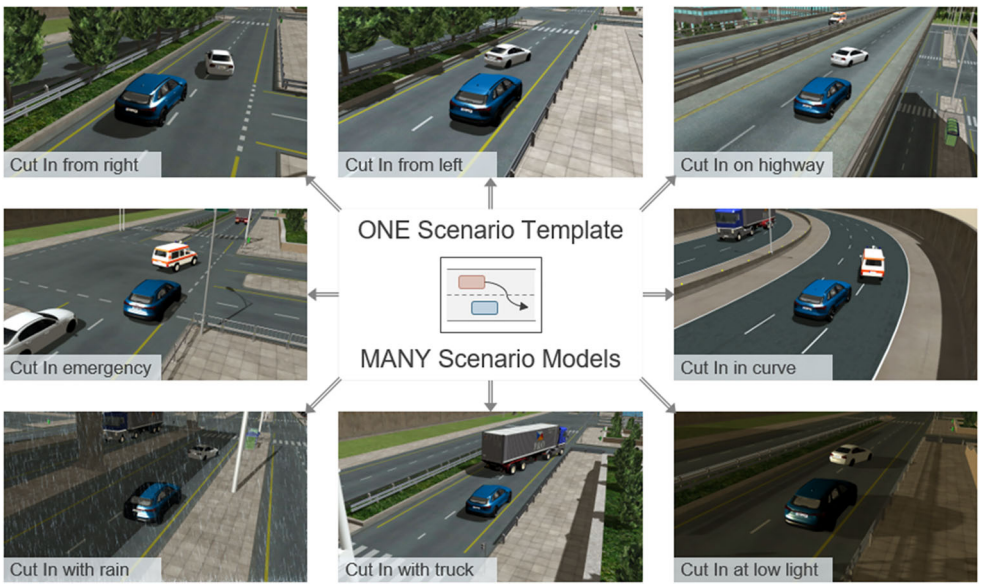

Fig. 1 Generation of virtual test scenarios based on a single scenario template (see [1])

[2] and OpenSCENARIO [3]. The SCIENIC language is a probabilistic programming language targeting the description and randomization of the static environment of a scenario. Therefore, parameters and parameter ranges are defined. The SCIENIC language is interfaced with various simulation tools, e.g. VerifAI [4] and CARLA [5]. The ASAM OpenX standards, on the other hand, emerged from the automotive industry and in addition enable the description of dynamic behavior. The standard consists of three distinct sub-standards: OpenDRIVE, OpenCRG, and OpenSCENARIO. The last sub-standard integrates the static description of the environment with a behavior description of the entities in the scene. A storyboard hierarchically describes the behavior of all entities, by using triggers and actions, allowing for the description of multiple concurrent actions. Currently, parameter variations are not integrated into the standard.

2.2 Overall system simulation infrastructures

Simulation technology and virtual methods become increasingly important in product development [6], virtual commissioning [7], as well as the verification and validation of technical systems [8]. The relentless advances in simulation technology are reflected by various simulation tools, e.g. the open source simulator for autonomous driving CARLA [5], Tesla's rendering and simulation-aided evaluation infrastructure for their AI-based autopilot [9], MSC's Virtual Test Drive [10], NVIDIA's factory floor simulation [11], and the open source traffic simulation package SUMO [12]. The developers of AI-based systems recognize this progress and the associated potential. Consequently, synthetic data and virtual tests are gradually being accepted and gaining importance.

However, the mentioned simulation tools are mostly designed for a specific application domain (e.g. autonomous driving), so their application in other domain contexts is difficult. To enable a cross-domain and cross-application simulation-based development of AI-based systems, the simulation technology must provide different perspectives to the system, resulting in a holistic convergence of the virtual and real world. This can be achieved through the systematic use of digital twins, as they represent a structurally similar replica of a real system.[1] Digital twins support transferability to real systems through their equivalent interface and enable the creation of so-called hybrid scenarios in which real and virtual components interact with each other in real-time. To unleash the full potential of digital twins, a comprehensive virtual testbed is required. In this work, we build upon the 3D simulation platform VEROSIM [14].

2.3 Processes for the validation of AI-based systems

Finally, it requires suitable processes to provide a framework for the application of available standards and technologies. Taking a look at classical approaches from e.g. software development reveals that mostly a formal specification of the system (i.e. the definition of subsystems and their coupling, interfaces, functions, etc.) is needed, to develop and validate the intended functionality. To evaluate the quality, performance, and fault tolerance of the software system approaches like code reviews, unit tests, and code walkthroughs are commonly pursued [15]. Transferring this type of established process to the development of AI-based systems brings some unique challenges, primarily because the functionality of the system is defined by data rather than a formal specification. Consequently, it is of great importance to examine and ensure the reliability and meaningfulness of the data used in the development process (i.e. evaluating the quality of data and avoiding data corruption [16]). In the automotive industry, the so-called SOTIF (ISO 21448 - Safety of the Intended Functionality [17]) faces similar challenges since it is designed for systems operating in a potentially unknown environment. The SOTIF approach especially attempts to minimize the number of unknown unsafe situations and thus connects to the problems in the development of AI-based systems, since data must cover all possible relevant operational scenarios. Additionally, it is reasonable to introduce a measure of uncertainties in the definition of quality metrics [18] and to continuously monitor the performance of AI-based systems in the development process to perform forensics in erroneous system states [19]. Despite the variety of loosely coupled approaches, a systematic and comprehensive approach for the simulation-aided development of AI-based applications does not exist yet.

2.4 Assessment of existing technologies

By now, a variety of technologies and concepts exist, which are needed as building blocks for the simulation-based development of AI-based systems. However, a toolbox that combines the individual building blocks into a comprehensive framework for the development of AI functionality has not yet been realized. This starts with the formal scenario description, which needs to be able to capture both the behavior description (e.g. OpenSCENARIO) and the formalized description of randomized scenario space (e.g. SCIENIC). In addition, relevant processes for the validation of AI-based systems (e.g. SOTIF) describe a basic procedure but do not provide

[1] The term digital twin has various facets and is sometimes defined very differently, depending on the respective context. In this paper, we use the terminology of [13] and use the digital twin as a virtual test object (simulation aspect).

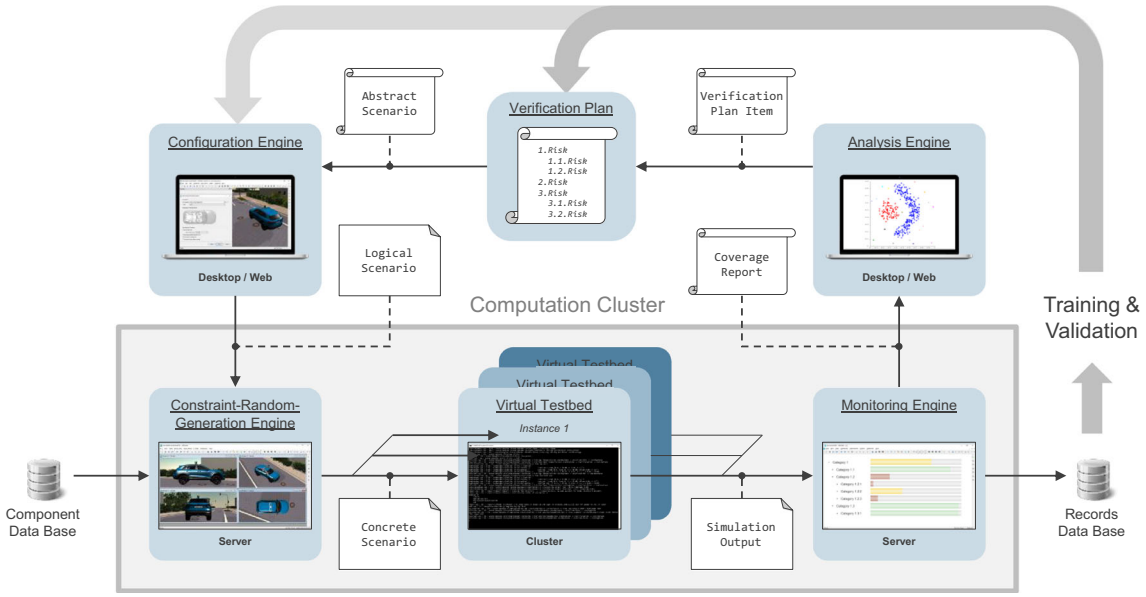

Fig. 2 Concept for systematic and target-oriented generation of synthetic data for the development and validation of AI-based systems with virtual testbeds (see [1])

any recommendations for the technical realization of the underlying process.

Based on the identification of the relevant technologies and processes, this results in a framework for the development of AI-based systems which fuses existing technologies, allocates them within a process, and enables a structured step-by-step validation of AI-based systems. The resulting architecture is a blueprint for the domain-independent view on AI development and thereby stands out from known domain-specific approaches. The structured consolidation of existing technologies is urgently needed and forms a cornerstone in the development of AI capabilities and is essential for the transfer of AI technology to complex applications and domains not yet conquered.

3 Architecture for the generation of virtual test scenarios

As already mentioned, the overall architecture has already been introduced in [1]. Therefore, this section briefly summarizes the existing parts and highlights the novel aspects, especially the data analysis and the feedback loop in the scenario generation process (see right side of

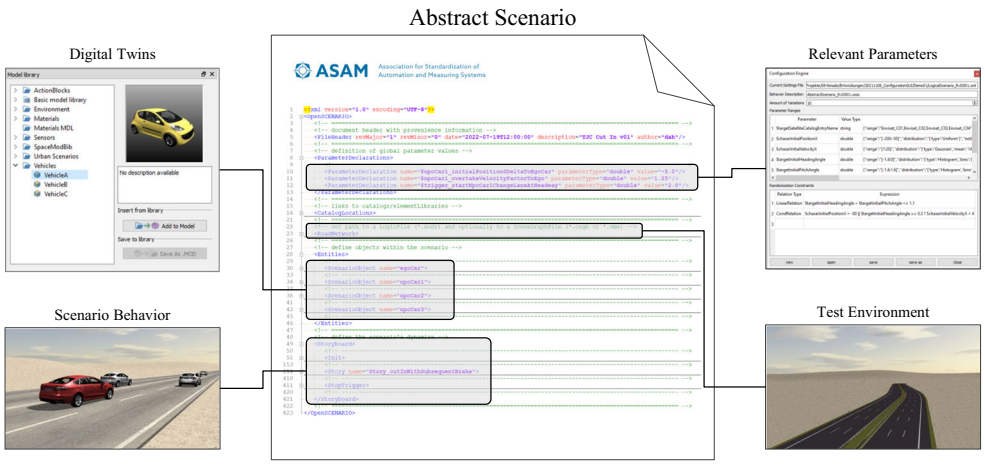

Fig. 3 Basic structure of the abstract scenario definition. All relevant variation parameters are defined as parameter declarations, and all scenario entities are linked to digital twins

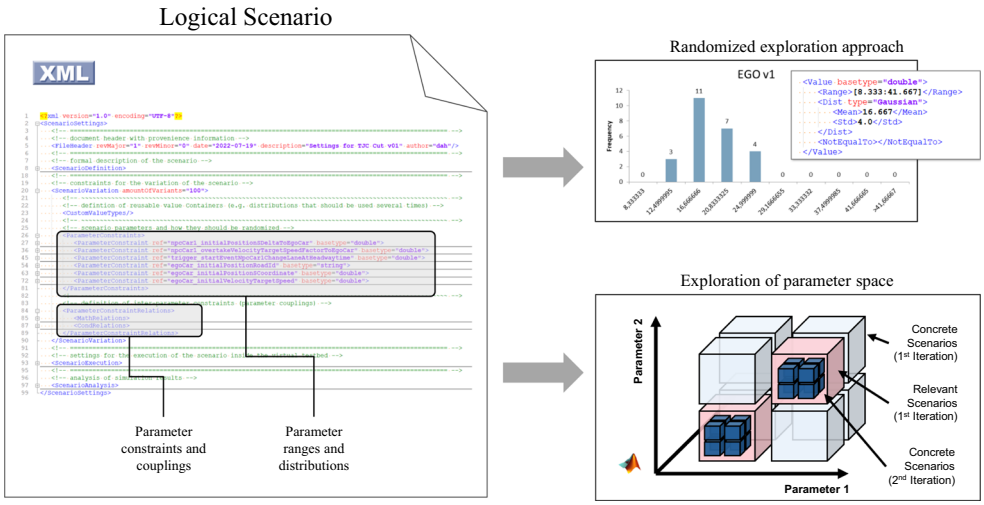

Fig. 4 Basic structure of the logical scenario definition. All parameters defined in the abstract scenario can be assigned with specific variation ranges. Based on this, the parameter space can be explored in a randomized or grid-based way

Fig. 2). The basic idea of the developed architecture is to iteratively generate scenarios, use them for simulation and data generation, and detect shortcomings and corner cases in the data set based on the analysis of these scenarios. Finally, these findings can be used to close the loop and generate additional meaningful scenarios for either training or validation purposes in the next iteration.

The entire process begins with the definition of an abstract scenario. The abstract scenario defines all relevant parameters, the required entities, the test environment, and the interactive scenario behavior. The abstract scenario is formalized according to the ASAM OpenSCENARIO standard (see Fig. 3). Once the abstract scenario is available, a configuration engine supports the generation of a logical scenario. The logical scenario spans the entire scenario space, by assigning parameter ranges and distribution to all variation parameters in the abstract scenario and thus envelops all relevant test cases. Additionally, the logical scenario enables the definition of couplings between parameters (linear equations and inequalities), constraining the scenario space (e.g. initial velocity of vehicle 1 needs to be greater than the initial velocity of vehicle 2). The logical scenario is the starting point for the exploration of the parameter space. Therefore, the concept provides two different methodologies: a randomization-based approach, and a uniform rasterization of the parameter space, see Fig. 4. The randomized exploration of the parameter space is especially suitable for large amounts of parameters, where the relevance of the individual parameters to the scenario is not obvious. This approach was presented in [20]. The rasterization approach uniformly samples the parameter space (see the cube in Fig. 4) aiming to iteratively identify interesting regions within the parameter space. Once relevant sub-spaces are identified, new scenarios are generated in these regions to generate additional relevant data. Within this contribution, we focus on this rasterization approach.

A randomization engine as implemented in [21] realizes both approaches and generates so-called concrete scenarios. These are concrete instances of the abstract scenario each with a unique set of values assigned to all parameters. Using the component database (a repository of digital twins), an executable simulation model can be generated automatically from each concrete scenario. The digital twins are combined with the scenario behavior, forming a virtual test scenario that can be used for data generation (Fig. 5). Details about the automated parsing and generation of the executable simulation model can be found in [22]. Impressions of the

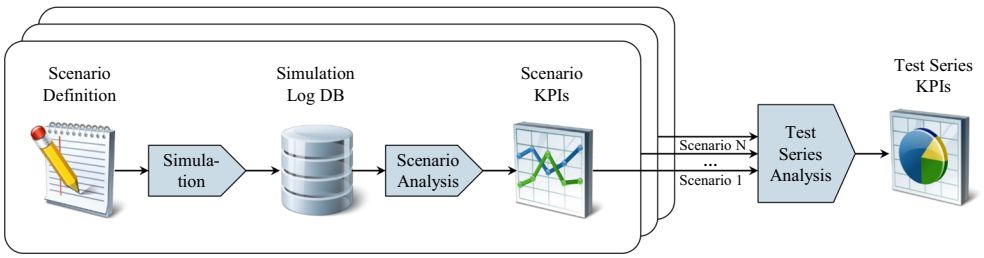

Fig. 5 Basic processing steps for the analysis of a test series. Each simulated scenario generates a simulation log database which is individually analyzed by applying appropriate scenario metrics. Once all scenarios have been analyzed, the entire test series can be evaluated based on the scenario KPIs

generated simulation models are shown in Fig. 6, which originates from the first application use case in the next section.

Once executable simulation models are available, the simulation can be carried out, either sequentially as well as parallelized on a computation cluster. Therefore, we developed a simple framework for the automated execution and scheduling of the individual simulation runs based on Python. Since we want to generate synthetic data from each simulation run, the data generation from the virtual test scenarios is of great importance. Therefore, we build upon a preexisting comprehensive logging functionality in the virtual testbed [23]. The applied logging approach generates comprehensive SQLite databases, providing a holistic replication of the entire simulation, which also can be used for simulation replays. To increase the performance of the simulation execution and reduce the amount of irrelevant data, we applied white-list logging, only storing the relevant data for the two use cases in the next section.

The data monitoring and analysis are realized on a newly developed Python framework. Figure 5 illustrates the basic steps for the analysis of a specific test series. The framework allows to query requests to the logging databases, and provides common data transformations and frequently used plotting functionality. Additionally, it provides modules to calculate key performance indicators (KPIs) based on common metrics. These KPIs and metrics are of great importance since they allow for the automated classification of scenarios. For example, if a distance metric between two vehicles becomes too small, this could indicate a critical scenario. Similarly, a collision between objects can be used as a classifier of an unsafe scenario. Based on this classification, interesting parameter sub-spaces can be identified and new scenarios can be generated subsequently. In addition, statistical couplings between parameters or scenarios can be analyzed, which can either highlight critical situations or indicate the risk of shortcut learning.

Since the amount of data to be handled easily becomes very large, it is clear that big data issues must be addressed during data-driven processes like this. The proposed architecture allocates these issues in the process chain and enables experts to develop targeted solutions here. However, the iterative scenario generation process and the flexible logging capabilities offer the possibility of reducing the amount of data generated so that the required application of specific big data technologies can be scaled as needed.

A fundamental design decision for the developed framework was a certain independence of the individual process modules. This way, the individual modules can be configured and combined for arbitrary use cases. By building upon the OpenSCENARIO standard and commonly used technologies such as SQLite or Python, compatibility with industrial applications is inherently given. Additionally, the individual modules can be replaced by various realizations, allowing for flexible customization of the toolchain. Data can be injected and gathered everywhere, making the entire

Fig. 6 Execution of the simulation scenario inside the virtual testbed showing different views on the cut-in process simultaneously

process of data generation transparent and allowing for a specific analysis of data at every step within the toolchain. The flexibility and versatility of the proposed framework should be demonstrated in the following applications.

4 Applications

4.1 Validation of an AI-based ADAS

The first application example comes from the autonomous driving domain and centers around the validation of an AI-based advanced driver assistant system (ADAS). As part of a research project, a so-called Traffic Jam Chauffeur (TJC) is being developed. The TJC is an ADAS that can be activated in a dense traffic situation to automatically control the vehicle. It autonomously ensures that the vehicle stays in its lane and maintains speed if the traffic situation allows. However, the intended operational situation is characterized by great complexity, since vehicles are driving close to each other and short-term changes occur (speed changes, lane changes, etc.) that require a quick response. The TJC must therefore be able to reliably detect the relative movement of surrounding objects and, if necessary, brake the vehicle to maintain a speed-dependent minimum distance to prevent rear-end collisions.

To support the development team, we apply the presented approach for system validation using virtual scenario tests. For quality assurance reasons, the validation tests are to be performed explicitly without knowledge of the internal structure of the ADAS. We, therefore, use a digital twin of the test vehicle in which the ADAS is integrated as a black box. In the context of an initial risk assessment, it is rated as particularly critical if another vehicle suddenly cuts in, since this abruptly reduces the distance to the object in front. In such a case, it must also be ensured that the AI in the ADAS recognizes the cutting-in vehicle and adjusts the Ego vehicle's speed to re-establish a suitable safety distance. To be able to evaluate the ADAS's functionality as early as possible in the development, the system should be tested within the vehicle context. The presented concept offers the possibility to perform virtual test series in simulation without the need for expensive test drives with real components.

For the identified risk of a vehicle suddenly cutting in, we first define an abstract scenario that considers three other vehicles in addition to the Ego vehicle, and one of them changes to the lane of the Ego vehicle. Figure 7 represents the initial situation graphically. In the beginning, the Ego vehicle is surrounded by the three NPC cars, and all cars are moving at the same speed. The positions and velocities of the NPC cars are defined relative to the Ego vehicle, making it very easy to move the whole test drive to a different position inside the virtual test environment. After two seconds, the NPC vehicle in the right lane accelerates and, as soon as it is ahead of the Ego vehicle for more than two seconds, it cuts into the Ego vehicle's lane. Depending on the distance and difference in velocity, this makes it necessary for the Ego vehicle to react. The scenario also includes a sudden deceleration of all surrounding vehicles to a complete stop a certain time after the cut-in process, a behavior that is not uncommon in a traffic jam. Therefore, at the end of the scenario, the TJC must also bring the Ego vehicle to a stop without hitting the vehicle in front. To keep the scenario generic and flexible, all characteristic parameters are declared as variation parameters.

The next step is creating a logical scenario for the first test series. For this purpose, the variation parameters in the abstract scenario are first evaluated for their relevance. The initial relative longitudinal offset d_1 of the cut-in vehicle to the Ego vehicle, the overtaking velocity v_2 of the NPC car 1, and finally the time when it starts the cut-in, which results from the headway time t_{headway}, are identified as particularly relevant. The logical scenario for the first test series therefore initially defines variation ranges only for these three parameters and assigns constant values to all other parameters. The used variation ranges are given in Table 1.

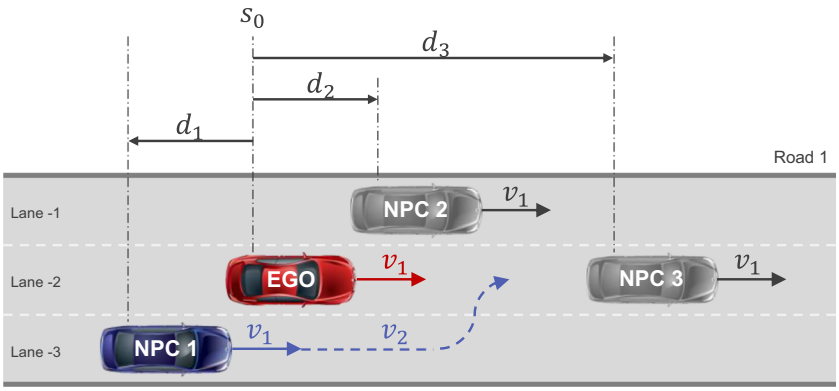

Fig. 7 Visualization of the specified abstract scenario for the automotive application example containing four vehicles on a road with three lanes

Table 1 Variation parameters for the first data set

Parameter name	Value range	Unit
d_1	$[-10 \ldots +10]$	Meter
v_2	$[1.1 \ldots 2.0]$	Relative factor to v_1
$t_{headway}$	$[0.5 \ldots 5.0]$	Seconds

The reduced dimension of the parameter space resulting from the estimation of parameter relevance allows the application of the approach for the systematic rasterized exploration of the parameter space. Therefore, in the logical scenario, equally distributed ranges for the three parameters are defined and corresponding concrete scenario variants are generated. Every concrete scenario is a specific OpenSCENARIO file with a unique parameter set for all parameter declarations. Thanks to the parser, each scenario can be selected directly in the virtual testbed and automatically translated into an executable simulation model when loaded. Figure 8 shows the resulting 3D model.

As mentioned earlier, we vary over three different parameters, spanning a three-dimensional scenario space. For the initial data set of test scenarios, all three dimensions are sampled uniformly. This results in 64 concrete scenarios for the first data set. With the help of the Python tool set, all scenarios of the first test series can be simulated automatically. Likewise, the logging of the simulation data is carried out automatically, forming the basis for the analysis of all scenarios.

The cube in Fig. 9 shows the sampling of the entire scenario space described by the logical scenario. Each point in the grid visualizes an individual simulation run. In total 64 simulations were carried out for the initial exploration of the scenario space and the evaluation is performed using the developed Python tools for analysis. To classify the criticality of the scenarios, the Time-to-Collision (TTC) metric is calculated in a post-processing step, taking into account the velocities of the individual vehicles and their bounding boxes. Therefore, the relative distance between both bounding boxes $x_1 - x_2$ and the relative velocity $v_1 - v_2$ of both vehicles are taken into account. The TTC can be interpreted as the predicted point in time where both bounding boxes intersect and thus both vehicles collide. Consequently, the TTC can be used to automatically measure the criticality of traffic scenarios. The lower it gets, the more critical a given scenario is. If both bounding boxes do not collide, the TTC is set to a chosen maximum value.

$$TTC(t) = \frac{x_1(t) - x_2(t)}{v_1(t) - v_2(t)} \tag{1}$$

If the TTC never drops below a given threshold of five seconds, the scenario is classified as a safe scenario. If the TTC drops below this threshold, the scenario is classified as critical, since it might cause discomfort to the driver. Collisions between vehicles ($TTC = 0$) are detected by collision signals generated by the simulator and state a severe failure of the system, which needs to be avoided.

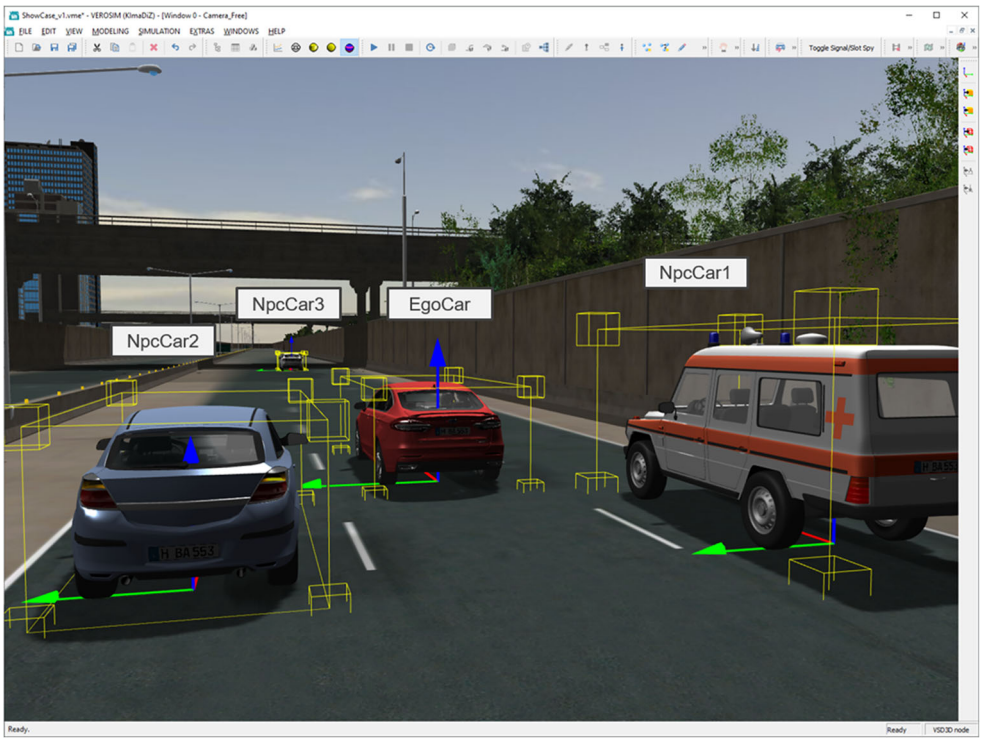

Fig. 8 Automatically generated executable simulation model for the automotive example

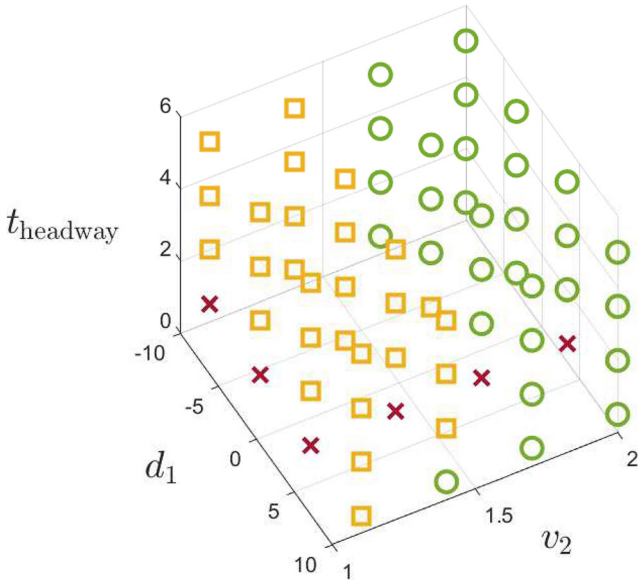

Fig. 9 Explored parameter space. Each point is assigned to a specific simulation run. Circles indicate safe scenarios, squares indicate critical scenarios and a cross indicates a scenario with a collision of the Ego vehicle

Taking a detailed look at Fig. 9 reveals two observations: First, there seems to be a separation plane between the safe and critical situations, which is defined by the relative velocity v_2. Second, the scenarios with collision form an "L-shape" in the d_1-v_2-plane. The first observation can be explained easily. If the relative velocity is big enough, the NPC vehicle will be safe in front of the Ego vehicle, regardless of the headway time t_{headway} and thus, a collision can be avoided. The second observation requires more sophisticated analysis. First of all, it is noticeable that all scenarios with a collision are characterized by a low headway time t_{headway}. The red (cross) scenarios with $d_1 =$ const. are scenarios where the NPC vehicle starts slightly in front of the Ego vehicle. Consequently, regardless of their relative velocity, the headway time t_{headway} causes the cut-in to start immediately after two seconds. Within this period, the NPC vehicle was not able to generate enough headway. The sensors used by the TJC are not able to detect the NPC yet, since it is not completely in front of the Ego vehicle. Subsequently, a collision occurs. The second line of red (cross) scenarios is characterized by very small relative velocities $v_2 =$ const. In combination with a small relative distance of d_1, this will have the same effect. The NPC will perform a cut-in without the Ego vehicle being able to detect the maneuver. Consequently, the TJC is not able to react accordingly and a collision is caused.

Collisions must be avoided. To get a detailed understanding of the transition from critical scenarios to a scenario with collision, we perform an in-depth analysis of a critical scenario in the direct neighborhood of a collision scenario. Figure 10 shows the TTC calculated in post-processing. It shows a significant indentation at 15 seconds. Comparing this to the velocity profiles, the change in TTC indicates the breaking of the NPC vehicle, once the traffic jam occurs. The Ego vehicle reacts properly, reduces its velocity, and keeps the desired safety distance. Nevertheless, another much more striking change in the velocity profile of the Ego vehicle can be seen, which does not correlate to the TTC. Here, the NPC cuts in really close to the EGO vehicle. To keep the desired safety distance, the TJC performs a hard-breaking maneuver that shows up strikingly in the velocity profile at 7.5 seconds. The TTC is not affected by this, since the velocity of the NPC is greater than the velocity of the Ego vehicle and a collision will not occur, even though both vehicles have a really small distance from each other.

Based on these findings, we are now able to generate new scenarios, especially targeting the small region between critical scenarios (yellow, square) and collision scenarios (red, cross). Additionally, to obtain more precise KPIs for the second iteration of the validation process, it is useful to consult additional metrics that take into account, for example, the distance between both vehicles while cutting in. This way, a multidimensional feature vector can be built over time, allowing for better classification of the scenarios, taking into account all relevant aspects of the use case.

4.2 Validation of AI-based rendezvous and docking maneuver

The second application example originates from the aerospace domain and deals with the automation of rendezvous and docking maneuvers (RvD). It is a standard maneuver for interacting space segments and can already be carried out automatically in parts. However, for safety and reliability reasons no AI methods have been used for it so far. New approaches based on AI must therefore prove their reliability in particular. However, realistic tests are extremely difficult to perform on earth. Therefore, virtual test scenarios provide a unique opportunity for the introduction of AI-based controls in spaceflight, and with the help of the methodology presented here, the training and validation data can be generated in suitable quantity and quality with reasonable effort. In this concrete example, we test and validate an RvD approach control system developed in the context of a research project. As in the previous example, we do not have insight into the internal structure of the control system to ensure neutral and unbiased validation testing.

The RvD maneuver usually consists of two consecutive phases. In the far phase, the chasing satellite is first roughly navigated into the space sector where the target satellite is located. Only when the chaser has reached the space segment it changes to the near-phase approach. The global

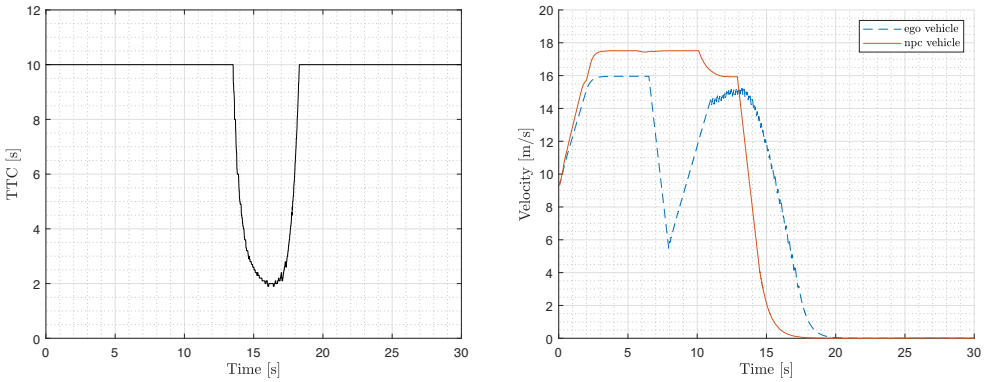

Fig. 10 Time-resolved analysis of a single critical scenario from the entire scenario space

(a) Time-to-Collision (b) Velocity profiles

coordinate guidance is deactivated and instead, the chaser starts to navigate in coordinates relative to the target using its local sensors. Therefore, in the context of the feasibility study described here, the chasing satellite activates a lidar sensor that captures 3D point clouds in a cone-shaped field of view with an aperture angle of 120 degrees. The AI-based onboard flight control system now tries to detect the target satellite within the captured point clouds and to approach it by commanding the corresponding actors (here thruster). At the end of the near approach, the chaser must have reached the near field of the target, i.e. only a few meters away from the target, and must have aligned its orientation to the target.

As part of a safety analysis, a risk is identified that the chaser will improperly estimate the position of the target satellite and, as a consequence, execute an inappropriate or, in the worst case, even dangerous approach trajectory. To cover this risk and to evaluate the current functionality of the controller system we define an abstract scenario for this near-phase approach. Figure 11 illustrates the basic structure of the scenario. It contains two satellite entities that are linked to the digital twins of the corresponding satellites. The digital twin of the chaser contains a virtual lidar sensor, virtual thruster actuators as well as the current version of the controller system (as a black box). The chaser has just completed the far approach phase and is therefore at the edge of the space sector where the target is located. We define an entry window in the yz-plane from which the chaser enters the near approach corridor. With the activation of the lidar and the control system under test, the near approach begins, in which the chaser automatically attempts to reach a suitable target position and orientation in the required safety distance based on the sensor data. The actual approach trajectory is therefore not determined in advance but results from the active attitude control.

It cannot be assumed that the orientation of the target to the input window is known at the beginning (the target might be uncooperative). However, it is likely to have an effect on object detection and position control within the control system. Therefore, it is defined as a variable parameter in the abstract scenario. Likewise, the exact entry point within the entry window is kept variable by two parameter declarations. This allows the creation of very different initial situations (approach from the right, left, diagonally above, etc.). Further parameters are kept constant for the first test series. Since the number of relevant parameters is still manageable, but their influence on the approach can hardly be estimated in detail, we choose the

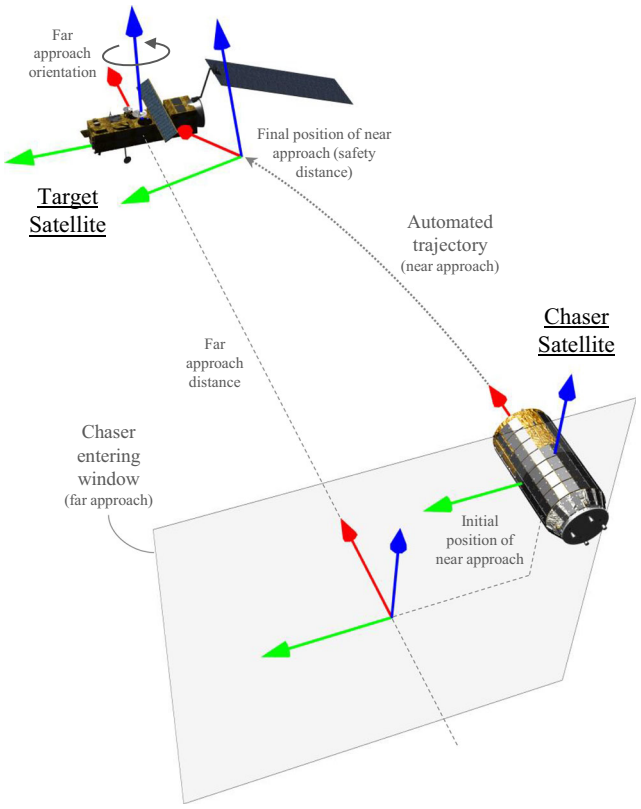

Fig. 11 Visualization of the specified abstract scenario for the aerospace application example containing a chasing satellite that tries to perform an automated near-phase approach to a target satellite

Table 2 Parameter ranges for the first data set

Parameter name	Value range	Unit
$y_{\text{chaser,init}}$	$[-10 \ldots +10]$	Meter
$z_{\text{chaser,init}}$	$[-10 \ldots +10]$	Meter
$\varphi_{\text{target,heading}}$	$[-1.571 \ldots +1.571]$	Radian

grid-based approach for the systematic exploration of the parameter space in this application example again. Thus, the next step is to derive a logical scenario that defines appropriate value ranges for the variable parameters of the abstract scenario. Table 2 lists the range definitions chosen for the first iteration.

The variation engine generates 45 different concrete scenarios for the first iteration, with each parameter sampled in an equally distributed manner. Since it can be assumed that the orientation of the target might have a particularly large influence on the test result due to its irregular shape, we sample the associated parameter with a higher resolution than the two position-related parameters. The generated concrete scenarios are then automatically simulated in the virtual testbed. Figure 12 shows the 3D view of the scenery of one concrete scenario. According to the scenario definition, the specified digital twins are included for the chaser and target and moved to the specified initial positions. At the start of the simulation, both the lidar sensor and the control system that controls 16 different thrusters are activated in the chaser.

It is possible to display different views simultaneously during the simulation to be able to observe the process from different perspectives and to qualitatively examine it manually. Figure 13, for example, shows three different perspectives: The topmost is that of an external observer. Here, the progress of the approach can be easily observed as the chaser satellite slowly navigates from the right side to the target on the left side. The two lower views, on the other hand, are from observers moving along with the chaser and give an insight into the perspective that the chaser has. Additionally, the 3D point cloud taken by the sensor is rendered as well (red points on the satellite).

The simulation is followed by the evaluation based on appropriate metrics and key indicators, which can be calculated automatically based on the logged simulation data. In the presented case, the scenarios are first classified based on their output. Therefore, a simple distance metric D was applied, comparing the final position of the chaser p_{chaser} to the position of the target satellite p_{target}.

$$D = \sqrt{\| p_{chaser} - p_{target} \|} \tag{2}$$

In addition, the duration of the approaching maneuver as well as the detection output of the sensor processing system were evaluated. Again, we used the simulator's collision signals to identify disastrous scenarios, where the chaser collides with the target. In combination, these four metrics form a feature vector that is used to classify the scenarios.

Figure 14 shows the classified parameter space for the first iteration cycle. The scenarios marked in green (circle) were successful, i.e. the chaser independently reached a suitable target position based on its sensors and actuators. The scenarios marked in gray (triangle) are those in which the approach exceeded a predefined maximum duration for the approach. The yellow (square) scenarios, on the other hand, are failures where the chaser was not able to even attempt an approach. The red (cross) scenarios indicate particularly critical failures in which the chaser collided with the target.

The distribution of the failures within the parameter space confirms the problem of estimating the success probabilities and the influence areas of the parameters in

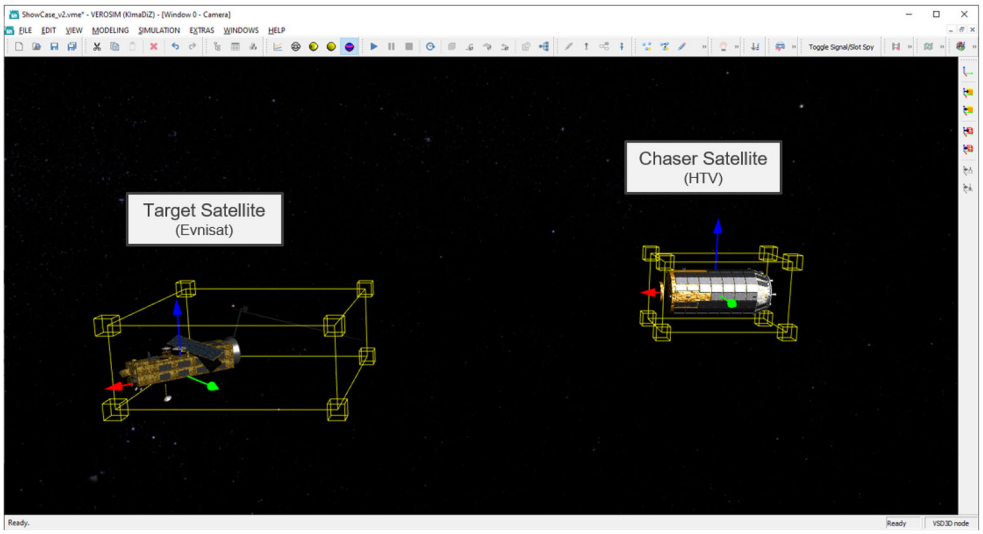

Fig. 12 Automatically generated executable simulation model for the aerospace example

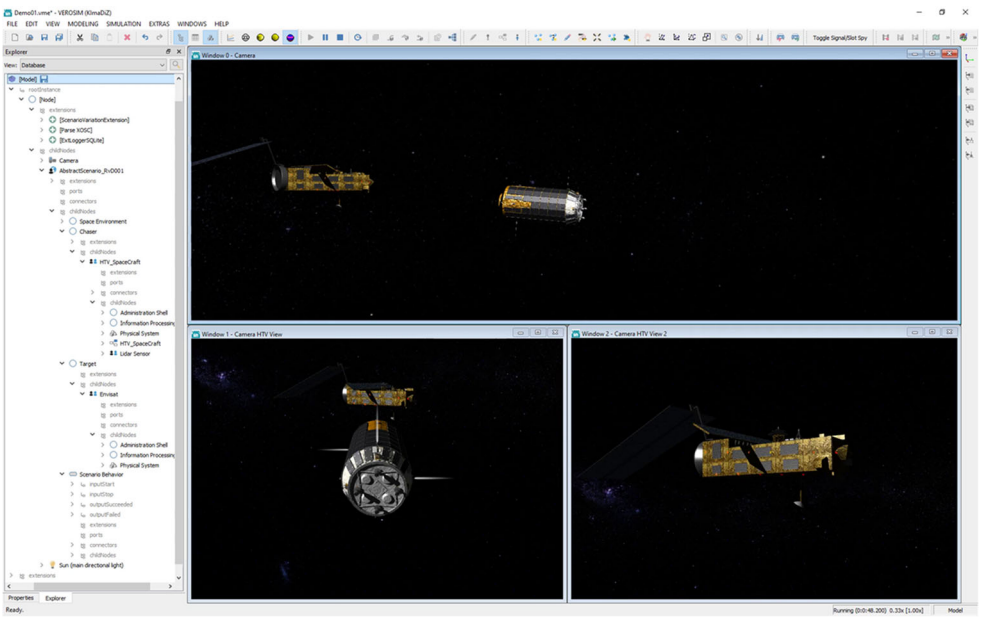

Fig. 13 Execution of the simulation scenario inside the virtual testbed showing different views on the near approach process simultaneously

advance. Let us first take a closer look at the scenarios in which the approach could not be started at all. It appears to be some coupling between the vertical distance between the chaser and target and the orientation of the target satellite (representing the orientation of the approach corridor). A more detailed examination of these scenarios reveals that the cause of the failure lies in the environment perception by the sensor system. The specific target satellite is characterized by a very large solar panel mounted inclined on the tail. The reflectivity of the laser light used by the lidar is much lower on the solar panel than on the gold coating that makes up most of the main body. In addition, the lidar has only a limited resolution concerning the point density of the generated point cloud. Therefore, the following

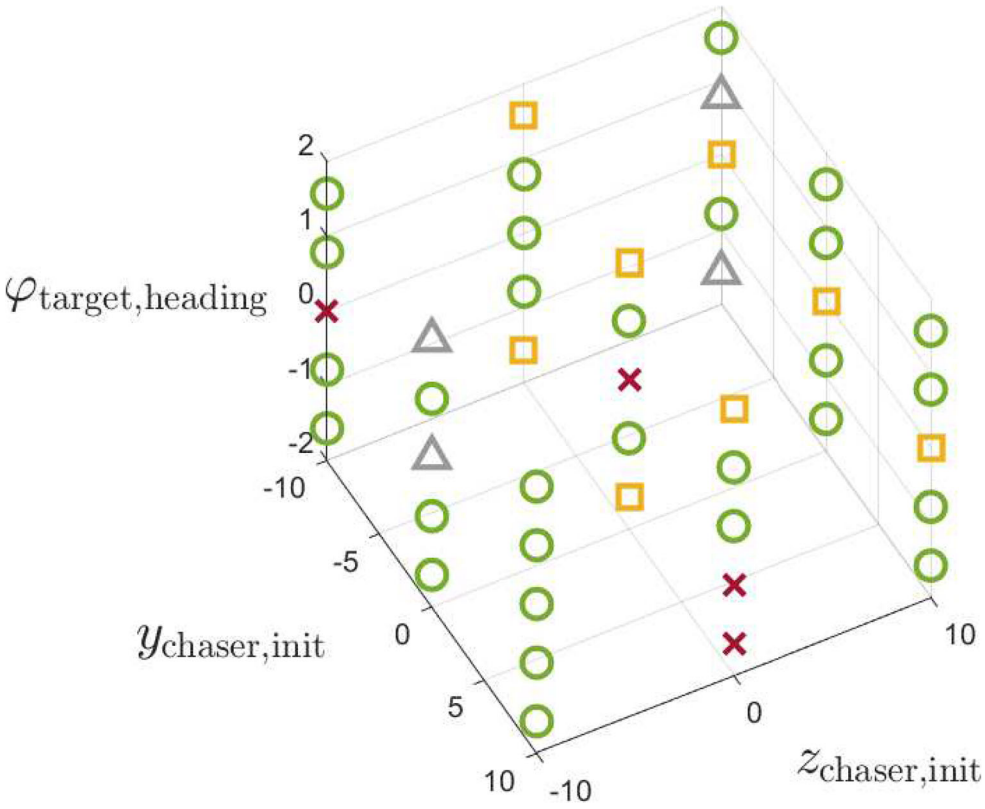

Fig. 14 Explored parameter space within the RvD use case. Each point is assigned to a specific simulation run. A circle indicates a successful RvD approach, a square indicates a scenario where the target was not detected, a triangle indicates a scenario where the target was not reached within the given time frame and finally, a cross indicates a collision between the chaser and target

problem arises in the above scenarios: The target satellite is (coincidentally) aligned in such a way that the easily recognizable main body is completely or to a large extent hidden by its solar panel so that the satellite is not properly recognized by the sensor due to the limited point density. Thus, the controller lacks the target and the chaser remains in its initial position and waits for the target to become visible. This example illustrates the need for physics-based overall system simulations since an idealized sensor that did not take into account the reflection characteristics of the materials would not have encountered this problem.

Much more critical, however, are the failures that resulted in a collision with the target satellite. This must be avoided under all circumstances, which is why the causes of the collision must be investigated in more detail (i.e. we need to perform scenario forensics). Again, the location of the corresponding scenarios in the parameter space does not allow for a quick interpretation. They all have in common that the collision occurs with the solar panel. In detail, however, two slightly different types can be identified, which have different causes. In the first type, the collision occurs frontally, in the other type at the side of the chaser. In the case of head-on collisions, the cause again lies in the way the lidar detects surrounding objects. The target satellite is again oriented with the solar panel facing the chaser. Nevertheless, the sensor also has enough hit points on the main body, which is why the approach is initially on schedule. However, the lack of detection of the solar panel over a long time leads to an overly optimistic distance estimation. Just shortly before the collision, the sensor also detects the solar panel, which leads to a sudden change in the distance estimation. Although the control system immediately initiates countermeasures, the collision can no longer be prevented due to the inertia of the system and the physical limitation of the control actuators, which is also reflected by the dynamic simulation model.

In the case of lateral collisions, on the other hand, the cause lies in the approach planning. The chaser approaches the target as planned, but does not take into account the side-mounted solar panel when aligning itself with the target. As a result, the targeted point for the approach may be below or too close to the solar panel, which leads to a lateral collision. A major problem here is that the lateral approach shortly before the collision is not visible in the sensor data and therefore no countermeasures are initiated up to this point. A course correction should have been made at a much earlier point in time.

Figure 15 shows the time series of the distance estimation for a successful approach (green, dot) and a failure with frontal collision (red, cross). In the case of the failure, one can see the already mentioned drop in the estimated distance after 58 seconds. The previous gaps in the plot also result from the solar panel. Due to the large coverage of the main body by the solar panel, it can temporarily lead to a loss of detection. The green curve on the other hand shows an example of the intended behavior. In the beginning, the

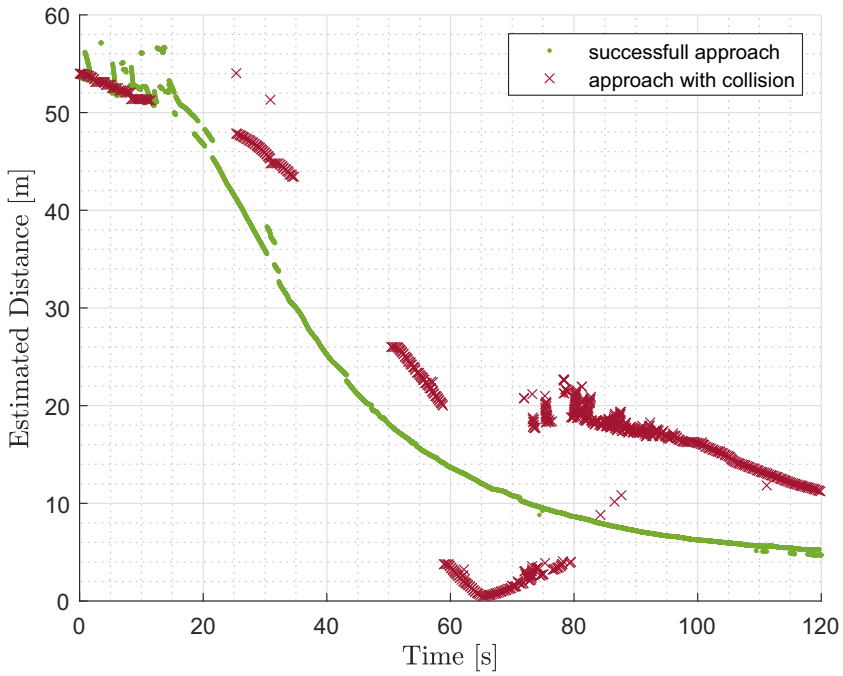

Fig. 15 Comparison of the estimated distance of the sensor for a successful RvD maneuver and an erroneous RvD maneuver with collision

measured distance decreases fairly fast, but the closer the chaser gets to the target, the slower the approach is until the safety distance is reached and maintained.

5 Conclusion

The validation of specific functionality is an important part of the development process for technical systems. This paper demonstrates the application of a structured process for validating technical systems based on simulated test series of the overall system in virtual operational scenarios. This scenario-based testing is particularly suitable for cases where the transferability of the validity of the individual system components to the validity of the overall system is very limited, and in cases where the internal structure of the system itself cannot be suitably validated. The latter is usually the case when parts of the system are realized with AI methods. The first case, on the other hand, occurs especially in complex and highly variable deployment environments, since they facilitate emergent behavior. However, a major problem that arises in black box testing is the usually vast parameter space. The presented process methodically addresses this problem with a monitored iterative approach for the successive exploration of the parameter space. This makes it possible to create structured trust in a developed functionality, which can hardly or only with difficulty be evaluated with classical analytical verification methods. Another advantage is the modularity of the individual process steps, which, thanks to the use of standardized formats, can in principle be flexibly exchanged and extended in their implementation. The basic applicability of the process was demonstrated by two feasibility studies from completely different application areas. However, these should be extended in the future by more extensive investigations taking into account the efficiency in terms of memory requirements and computing capacity. Furthermore, the current reference implementation of the process is semi-automated and currently requires expert knowledge at various points. Consequently, another goal for the ongoing development of the approach lies in the further automation of the process.

Funding Open Access funding enabled and organized by Projekt DEAL. This work is part of the project "KImaDiZ", supported by the German Aerospace Center (DLR) with funds of the German Federal Ministry of Economics and Technology (BMWi), support code 50 RA 1934.

Data Availability The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of Interests Ulrich Dahmen and Tobias Osterloh declare they have no financial interests. Jürgen Roßmann is managing director of the VEROSIM GmbH.

References

1. Dahmen U, Osterloh T, Roßmann J (2021) Generation of virtual test scenarios for training and validation of AI-based systems. In: IEEE international conference on progress in informatics and computing (PIC), pp 64–71
2. Fremont DJ, Dreossi T, Ghosh S, Yue X, Sangiovanni-Vincentelli AL, Seshia SA (2019) Scenic: a language for scenario specification and scene generation. In: Proceedings of the 40th ACM SIGPLAN conference on programming language design and implementation, pp 63–78
3. Association for Standardization of Automation and Measuring Systems ASAM OpenSCENARIO® Standard 29.07.2022. https://www.asam.net/standards/detail/openscenario/
4. Dreossi T et al (2019) VerifAI: a toolkit for the formal design and analysis of artificial intelligence-based systems. In: Dillig I, Tasiran S (eds) Computer aided verification (CAV), vol 11561. Springer. https://doi.org/10.1007/978-3-030-25540-4_25
5. Dosovitskiy A, Ros G, Codevilla F, Lopez A, Koltun V (2017) CARLA: an open urban driving simulator
6. VDI-Gesellschaft Produkt- und Prozessgestaltung (2004) VDI 2206: design methodology for mechatronic systems
7. VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (2016) VDI/VDE 3693: virtual commissioning
8. European Cooperation for Space Standardization (2009) ECSS-E-ST-10-02C – verification
9. Tesla AI Day Presentation Tesla, Inc., 29.07.2022. https://www.youtube.com/watch?v=j0z4FweCy4M
10. Fan Y, Krishnan K (2019) Whitepaper: open standards - essential for self-driving? Validating autonomous vehicles through billions of virtual test miles, 29.07.2022. https://d10tcz9jtwksbg.cloudfront.net/wpcontent/uploads/2020/02/Open-Standards-Essential-for-Self-Driving.pdf
11. Makoviychuk V et al (2021) Isaac gym: high performance GPU based physics simulation for robot learning. In: Proceedings of the conference on neural information processing systems (neurIPS 2021) track on datasets and benchmarks. https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/28dd2c7955ce926456240b2ff0100bde-Paper-round2.pdf

12. Alvarez Lopez P et al (2019) Microscopic traffic simulation using SUMO. In: 2019 IEEE intelligent transportation systems conference (ITSC), pp 2575–2582
13. Dahmen U, Roßmann J (2021) "What is a digital twin – a mediation approach". In: 2021 IEEE international conference on electro information technology (EIT), pp 165–172
14. Roßmann J, Schluse M, Schlette C, Waspe R (2021) Control by 3d simulation–a new erobotics approach to control design in automation. In: International conference on intelligent robotics and applications, pp 186–197
15. Technichal Committee (2011) ISO/IEC JTC 1/SC 7 software and systems engineering: ISO/IEC 25010:2011 systems and software engineering: systems and software quality requirements and evaluation, (SQuaRE), System and software quality models
16. Holland S, Hosny A, Newman S, Joseph J, Chmielinski K (2020) The dataset nutrition label: a framework to drive higher data quality standards. In: Hallinan D, Leenes R, Gutwirth S, De Hert P (eds) Data Protection and privacy: data protection and democracy (computers, privacy and data protection. Oxford, pp 1–26). https://doi.org/10.5040/9781509932771.ch-001
17. Technichal Committee ISO/PAS (2019) Road vehicles: safety of the intended functionality (SOTIF)
18. Hamada K et al (2022) Guidelines for quality assurance of machine learning-based artificial intelligence. SEKE:335–341
19. Solanke A (2022) Explainable digital forensics AI: towards mitigating distrust in AI-based digital forensics analysis using interpretable models. In: Forensic science international: digital investigation, vol 42. https://doi.org/10.1016/j.fsidi.2022.301403
20. Osterloh T, Dahmen U, Roßmann J (2022) Automated generation, execution, and evaluation of virtual test series. In: IEEE international conference on computational intelligence & virtual environment for measurement systems and applications (CIVEMSA)
21. Maqbool O, Roßmann J (2022) Formal scenario-driven logical spaces for randomized synthetic data generation. In: MODELSWARD
22. Dahmen U, Osterloh T, Roßmann J (2022) "Modeling operational scenarios for simulation-based validation of technical systems". In: IEEE international conference on artificial intelligence and computer applications (ICAICA)
23. Atorf L, Roßmann J (2018) Interactive analysis and visualization of digital twins in high-dimensional state spaces. In: 15th international conference on control, automation, robotics and vision (ICARCV)

Ulrich Dahmen received M.Eng. degree in electrical engineering with focus on information technology and environmental engineering at Hochschule Niederrhein University of Applied Sciences in 2016. Currently he is doing his PhD at the Institute for Man-Machine Interaction (MMI) at the RWTH Aachen University as a research assistant. His research focuses on simulation-based verification and validation of technical systems from the space, automotive and environmental sectors based on digital twins as virtual prototypes.

Tobias Osterloh received M.Sc. degree in electrical engineering with focus on system technology and automation of RWTH Aachen University in 2015. Since 2016 he is research assistant with the Institute for Man-Machine interaction investigating novel methodologies for the dynamic simulation of digital twin. He coordinated the MMIs subprojects within the DLR-funded joint projects iBOSS-3 and KImaDiZ and was named group lead for aerospace projects in 2020.

Jürgen Roßmann received the Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering from the Technical University of Dortmund, Germany, in 1988 and 1993, respectively. He was the Group Head with the Institute of Robotics Research (IRF), Dortmund, Germany, and was appointed as a Visiting Professor with the University of Southern California, in 1998. He returned to IRF as the Department Head and in 2005, founded the Company EFR-Systems GmbH. Since 2006, he has been a Full Professor and the Director with the Institute for Man-Machine Interaction, RWTH Aachen University, Aachen, Germany. Also in 2006, he was appointed the Deputy Director for the DLRs Institute for Robotics and Mechatronics. Prof. Rossmann was the recipient of several national and international scientific awards. He is a member of the National Academy of Science and Engineering, Germany.