

Simulation-Enabled Methods for the Development, Testing and Validation of Cooperative and Automated Vehicles

Sven Hallerbach¹
Ulrich Eberle²
Frank Koester¹

Abstract

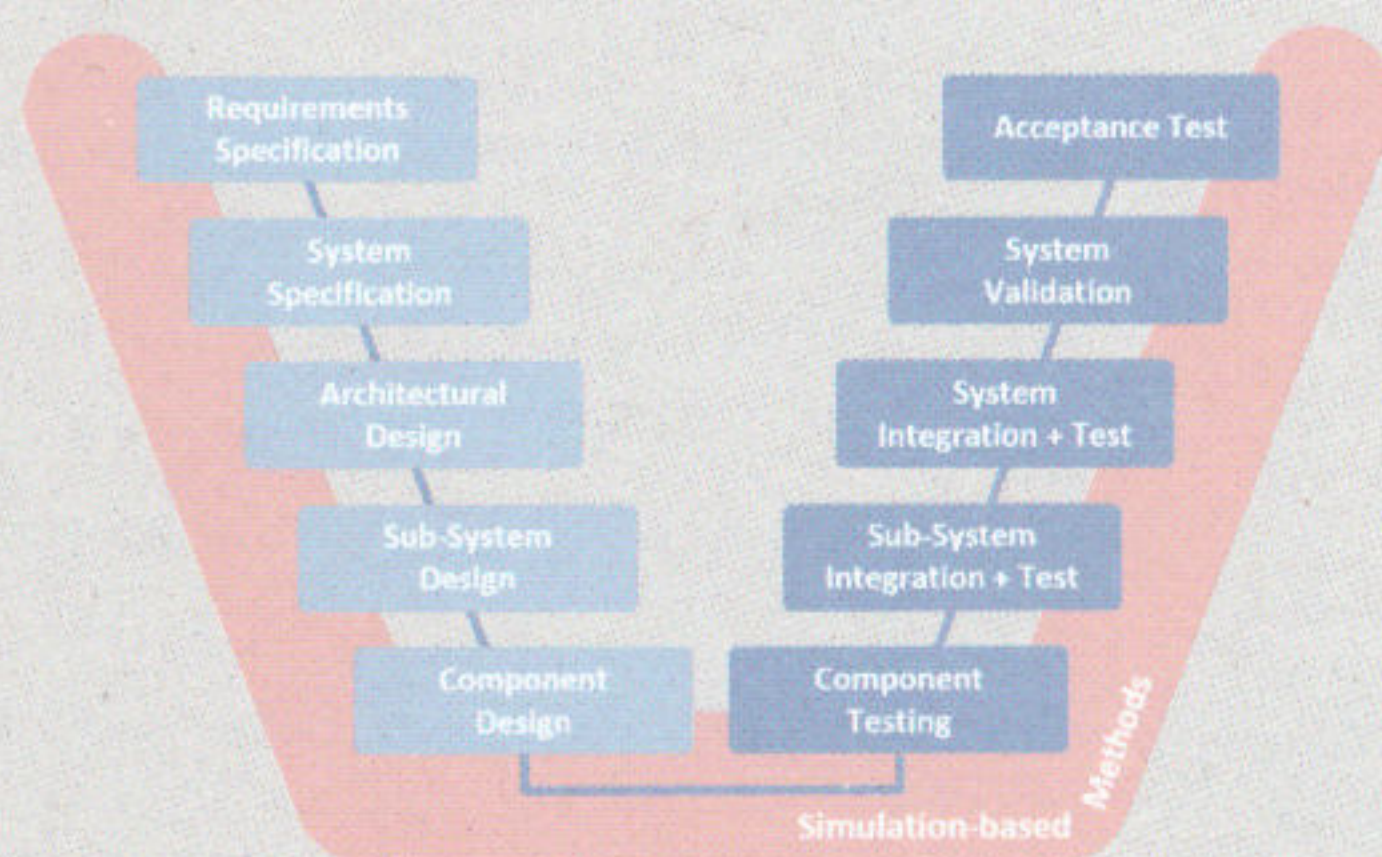
The release and public deployment of cooperative and automated vehicles remains a major challenge for the automotive industry. Despite the recent progress made by worldwide initiatives in this field, including the PEGASUS family of projects, demand for comprehensive simulation-enabled development, test and validation methodologies for cooperative and automated driving continues to rise. In addition, the number of applications can be expected to grow as well. Integrated AI components in particular are inherently complex systems that require automated, simulation-enabled and standardized methods and tools throughout the development and application chain.

The procedures need to be automatically set up and configured in such a way that they allow for an “as a service” approach. For this reason, an open testing architecture that operates similarly to a “bus system” using standardized components and interfaces is proposed. This represents a step towards a plug-and-play testing approach that supports the independence and exchangeability of components, systems-under-test, operational environments, scenarios, etc. Thus, an open testing architecture of this kind can be optimally utilized for scenario-based testing, which currently seems to be the best way forward in ensuring the safety of cooperative and automated vehicles. Development and testing across the entire spectrum of simulation-enabled methods, such as Model-in-the-Loop (MiL), Software-in-the-Loop (SiL), X-in-the-Loop (XiL) and Prototype-in-the-Loop (PiL), are supported up to non-virtual testing on proving grounds and traffic systems.

1.1 Introduction

Simulation-based procedures, methods and tools/toolchains are extremely important for a wide range of engineering tasks during the development of automated and connected vehicles and AI components (cf. Figure 1.1).

Figure 1.1:
Simulation-based
procedures,
methods and tools
for a wide range
of engineering
tasks [1]



Thus, it is possible to deal with the inherently complex advanced AI-based technologies, the wide range of functionalities as well as challenging environments, e.g. urban areas equipped with intelligent traffic infrastructures and communication technologies and also increasingly important backend systems. With a focus on behavioural

¹ German Aerospace Center (DLR)

² Opel Automobile GmbH

and dynamic aspects respectively, a large number of traffic scenarios and various traffic participants, in particular cyclists and pedestrians, who often exhibit non-normative or almost entirely unpredictable behaviour, have to be considered.

Along with these tasks, a wide range of users have to be supported by simulation-based procedures, methods and tools. One single tool is generally incapable of fulfilling their diverse needs and objectives. It is essential to couple or integrate different tools. This is where the idea of "simulation as a service" comes in.

Our article explores these topics and puts forward a concrete approach to organizing such methods and toolchains that benefits particularly from already existing standards. This approach is further refined and demonstrated for applications and engineering tasks that are currently of high practical relevance, such as developing and testing cooperative and automated vehicles and AI components.

1.2 Applications

Automated driving is one of the biggest and most inspiring engineering challenges of our age, since it combines automotive technology, conventional robot approaches and artificial intelligence-based methods in one complex safety-critical system, namely the automated driving system-equipped (ADS-equipped) vehicle. Simulation-enabled technologies already play a highly important role in

each of the underlying basic technologies, and will only become increasingly important going forward.

Applications for these approaches can be roughly structured as follows:

- Rapid prototyping of early development-phase AD systems
- Training AI-based algorithms and agents in virtual environments
- Analyzing the working mechanisms of AI algorithms based on external stimulation
- Scenario-based validation of AD systems
- Advanced X-in-the-Loop systems / hybrid reality in order to bridge the sim-to-real gap

AD systems pose the formidable challenge of a human driver no longer being available as a backup to the automation system, and this is the major distinction between an automotive AD system (SAE Level ≥ 3) and other similar automation applications in transportation, e.g., autopilot systems in civilian aviation. Here, expert users in the form of the pilots are constantly monitoring the environment and the system. Furthermore, Air Traffic Control is also monitoring the airspace.

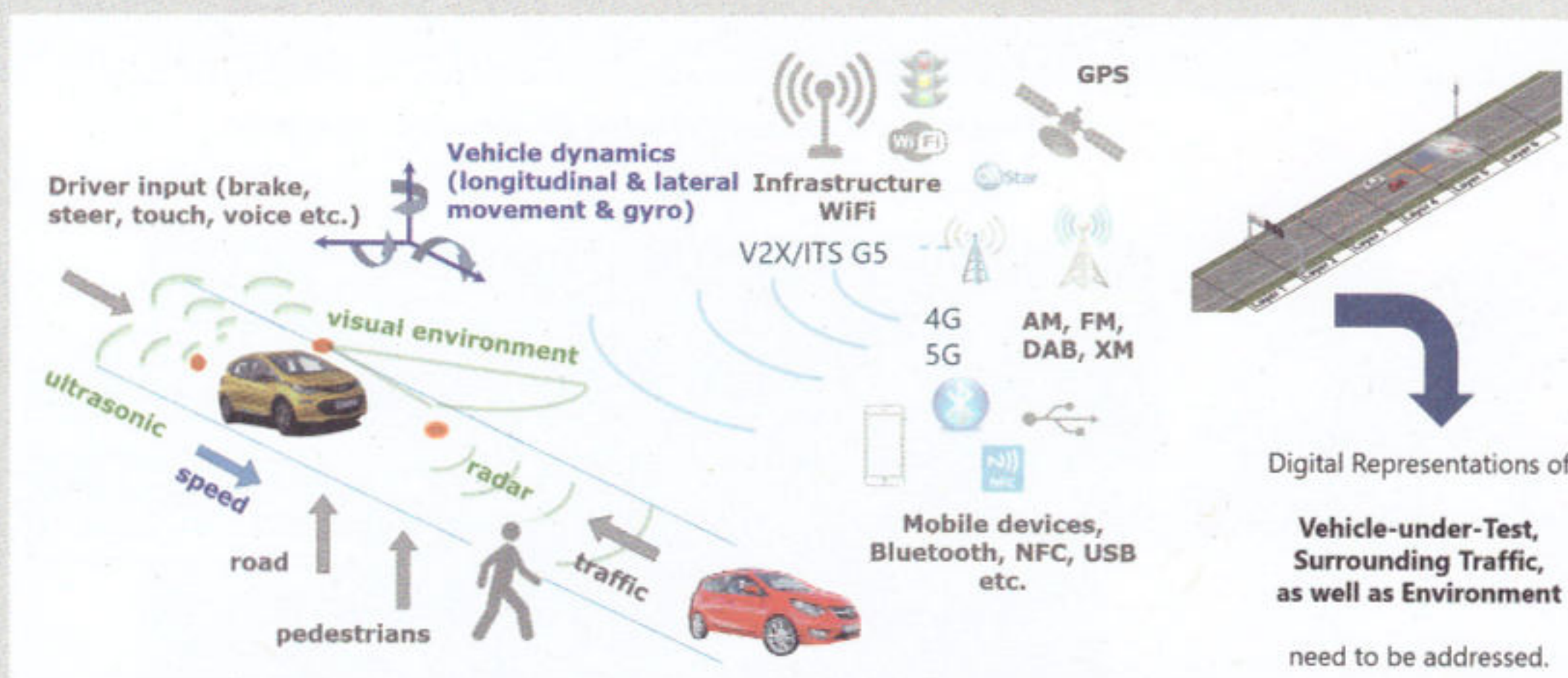


Figure 1.2: Digital twins and digital representations for virtual development and testing [2]

In terms of automotive applications, particularly in the urban environment, the technological challenge is the "open world" or "open-context", meaning it is neither sufficient nor even possible to work with a small, limited set of scenarios for the development and validation of the automated system.

There is a need to establish an approach for exploring the parameter space of the "unknown unknowns". As Figure 1.2 shows, this requires not only digital representations of the vehicle-under-test, but also the surrounding traffic and environment. Furthermore, a methodology benchmark needs to be identified. Projects within the PEGASUS family of projects decided to adopt the "GAMAB" principle, which means the new technology needs to be at least as good as the established technology. For the automotive community, that means that the AD system needs to be significantly better than an average human driver in a state-of-the-art modern car. Since humans only cause a severe accident every 12 million km on Germany's high-speed motorways for example, statistical considerations dictate that billions of km need to be driven to demonstrate a good performance level with a sufficient confidence level and test probability. In the urban context, the situational complexity is several orders of magnitude greater than on motorways. Finally, an automobile is a safety-critical system that could endanger occupants and other traffic participants if it is not properly designed and implemented.

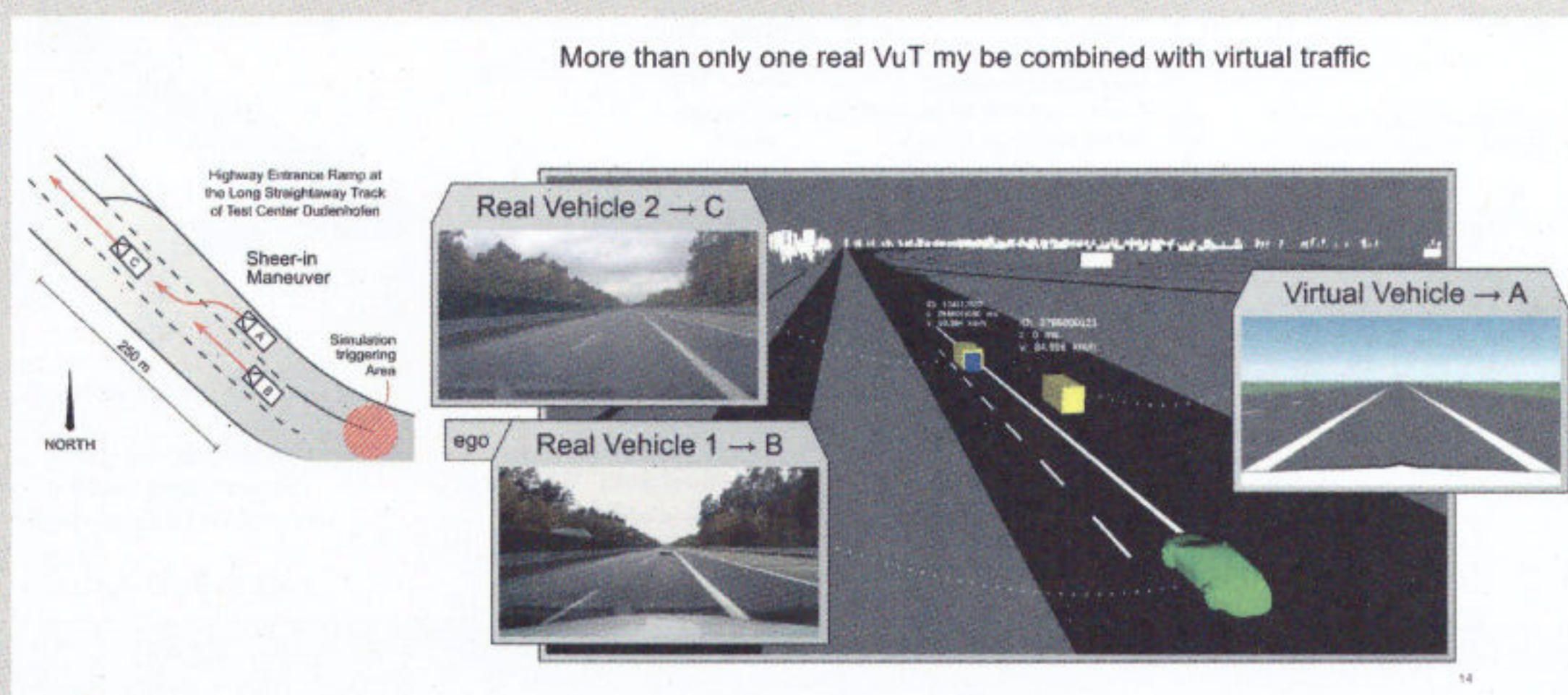
For this reason, it is not possible to test or train an early-stage AD algorithm on public roads. Rapid prototyping in a fully simulated environment is therefore of utmost importance and significance, since any flaws in the system will not harm the test driver in the vehicle-under-test or other traffic

participants. This holds especially true when cooperative and V2X features need to be investigated alongside automated functions. A cooperative feature such as cooperative highway merging takes place in a complex traffic environment and requires at least three cooperating vehicles and many more non-cooperating surrounding vehicles, as Figure 1.3 shows. Staging such an experiment by orchestrating test drivers is virtually impossible, not reproducible and, in terms of exploring system limits, also dangerous. For this reason, simulation-based approaches comprising multi-ego capability (i.e. several vehicles-under-test operating at the same point of time) and agent-based testing are key.

Similarly, using a pure real-world approach, it is impossible to responsibly train an AI algorithm (regardless of whether perception or path planning is considered, for example) since crashes will occur during the training stages and learning phases. This is of particular importance when self-supervised or even reinforcement learning AI algorithms are applied, which (by definition) learn from previous failures to achieve more mature stages. It could be assumed that first-stage training takes place in a fully virtual environment as shown in Figure 1.4 and moves up to the next stage of the "road to reality" when pre-defined performance and quality metrics have been attained.

A simulation environment can also be used to deliver external stimuli to a "black-box" neural network in a controlled and reproducible manner for the purpose of analysing the network's operational principles. This is particularly interesting since neural networks are often trained on and learn from data set characteristics that might not at first sight be apparent to a human analyst. By providing such external stimuli – from a traffic simulation,

Figure 1.3: Simulation and hybrid reality for developing and testing automated and cooperative features [3]



Key Role of Simulation and X-in-the-Loop → Incremental Steps

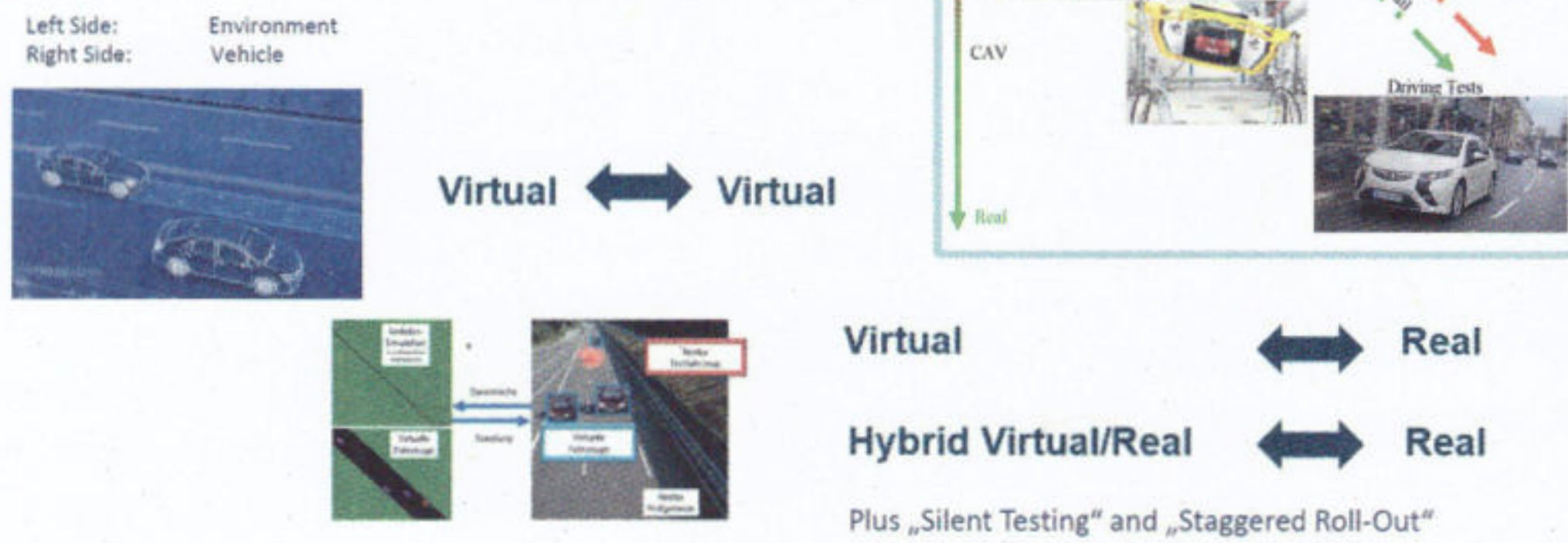


Figure 1.4: Incremental steps from MIL/SIL approaches over advanced Vehicle-in-the-Loop and Prototype-in-the-Loop/hybrid reality to public road operation [2]

for example – and by careful and controlled variation of the network combined with subsequent sophisticated analysis, valuable and highly relevant insights can be gained. By contrast, such a virtual environment could also be used to simulate adversarial attacks on an AI-based AD system in the controlled virtual environment. Doing so, and letting an adversarial agent compete against the AD module, could achieve a more robust final AI system.

The primary focus application of simulation-based approaches to automated driving involves validating a specific AD feature. Typically, there is a collection of simple scenarios from a process, such as an expert peer review or scenarios captured from real-world-driving, that are curated for use in a virtual environment. The commonality of both approaches is that they typically work using “scripted scenarios”, meaning that traffic participants are not independent agents, but follow scripted trajectories. This approach is an essential basis and has merit of demonstrating and measuring both the performance level of an AD system and reproducibility. To increase the long-term applicability of such scenarios, however, there is a need to transform the scripted traffic participants into independent agents capable of running in a long-term or higher-complexity scenario, such as approaching or emerging from a traffic jam, where the traffic is characterized by a large number of interacting traffic participants and the phenomena that result from these interactions.

Finally, simulation-based approaches will always address at least three distinctive parts

- Perception
- Path planning
- Vehicle dynamics

To reduce the complexity of the challenge of testing and tackling parameter space explosion, it is sometimes helpful to define procedures and tests that focus solely on a specific sub-field in a controlled manner. For example, full real-time physical sensor models are not required to test path planning and the actors/vehicles dynamics. Similarly, a full and highly realistic vehicle model is not required to test perception. Simulation-based approaches that exclusively address perception could, for example, augment existing data captured from real-world scenarios by inserting additional dynamic or static objects or by changing certain features of the specific scenario, such as brightness levels or other weather conditions. This is particularly important where the augmented scenario is not accessible from real-world driving since it occurs too infrequently to be captured or would be too dangerous to provoke.

Such focused simulations may be combined to form a systems approach based on the establishment of regions of interest based on sensor range. Within this region of interest, the world would be simulated in high fidelity, and outside this moving region or window around the vehicle-under-test, a computationally less demanding lower-fidelity tool could be used to simulate the environment, vehicle physics and agents. Since the operational im-

plementation and documentation are also of utmost important, a powerful parameter variation and test execution tool is an indispensable part of the complete toolchain.

To address the aforementioned challenges, a simulation-based toolchain for automated driving needs both to be highly modular and flexible and to cover different levels from MiL to hybrid reality approaches on a proving and even to real-world driving. It must also integrate scripted and agent-based scenarios or even a combination of both. Additionally, the toolchain should tackle the early phases of the development process as well as the late validation parts and should be compatible with all phases from MiL to Prototype-in-the-Loop/hybrid reality by applying shared definitions, metrics, simulation models and scenario libraries. Our approach to using an open and modular approach to achieve these targets is described in the following chapters.

1.3 Framework

Simulation has become an important and powerful tool with a high degree of relevance to a wide range of engineering tasks, ranging from the identification of first ideas and requirement elicitation through to final testing and assessment. During the systems operation and maintenance phases, simulation can also serve as a tool for improving our understanding of abnormal or non-normative situations, for example, and deriving as well as assessing mitigative actions or functional improvements. The DevOps idea and related processes cover, among other things, these tasks and phases. In general, the entire product lifecycle is covered (cf. Figure 1.5). DevOps integrates activities from systems design and development with sys-

tems operation, the general intention being to reduce the time needed to develop the system and ensure high system quality while providing continuous delivery based on field data and experience. While DevOps has its roots in IT systems and software development, it is now used in other areas, including the automotive industry.

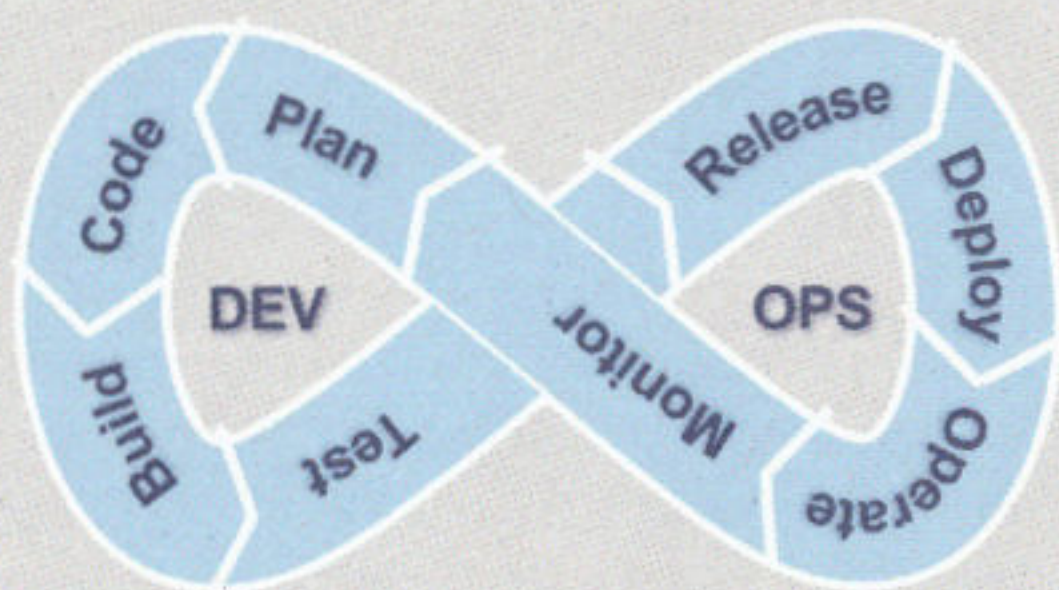
The main phases of the DevOps cycle are briefly described below:

- **Plan** – Collect and precisely describe requirements, define the architecture and other aspects relevant to the implementation/construction of the system and its release.
- **Code** (= implement or construct) – Implement and/or construct the component parts of the system.
- **Build** – Integrate the various components and build the system
- **Test** – The system undergoes testing (over and above the continuous testing conducted during previous phases and, in particular, end-of-line tests).
- **Release** – The system is available for deployment, all relevant key performance indicators (KPIs) have been adequately fulfilled.
- **Deploy** – The system/product can be made available and deployed for customer use.
- **Operate** – The system/product is observed during use by customers. Further KPI compliance can be verified during operation using field data.
- **Monitor** – The long-term performance of the system/product can be monitored during operation. The findings from this phase indicate where improvements or new features are needed. Improvements to system production itself can also be identified (by means of new or extended test catalogues or adapting KPIs to incorporate customers' justifiable environmental needs).

1.3.1 Open testing architecture

An open testing architecture (OTA), see Figure 1.6, was briefly proposed by Koester et al. in the ASAM SIM:Guide [4] and shortly after that at the SETLevel interim event [5]. An OTA consists of generic

Figure 1.5:
Phases typically
covered by the
Development Op-
erations (DevOps)
cycle



components and assumes a high level of standardization for components and their connecting interfaces. Utilizing standardized components, interfaces, data structures and architectural design ensures a quality standard that is based on a consensus between participating players, such as OEMs, TIER1s, etc. Furthermore, this way of designing the OTA ensures flexibility, configurability and exchangeability. This seems to be what is missing when user demand at this point in time is analysed. Moreover, the code, control algorithms and perception modules inside the standardized components can be kept private and thus remain the intellectual property of the developer. This ensures flexibility while still delivering an agreed overall standardized framework.

The OTA resembles a bus system where various components communicate with each other in a predefined manner. Bus systems connect a variety of components that serve different purposes and enable them work together through a standardized, interface-based form of communication. Like a bus system that amalgamates electronic control units (ECUs) into a fully functioning end-to-end system, the OTA connects components that possess a broader variety than just ECUs, such as simulation models, operational environment descriptions, entire test descriptions, etc.

Configuration management serves as a continuous approach to set up the individual components while providing an overall configuration op-

tion that can be monitored and, if required, reconfigured at any time. This scheme is based on the **standardized architecture**, which predefines both the minimum and the essential setup options and additionally limits the possible configurations to a certain extent, depending on the components chosen.

A **standardized architecture** provides several advantages, including a limit on the minimum number of configurable components that does not restrict the complexity of the components themselves, yielding a finite set of components that should be sufficient for the intended aim. In addition, standards-based architecture necessitates standardized communication between components. The communication system, which consists of **standardized interfaces & data structures** serves to ensure proper communication between components and provides a further necessary standard for setting up each individual component with respect to their interfaces and data structures. Thus, the entire OTA communicates in a predefined standardized way and utilizes data structures which are composed in the same manner.

The **specify (test)** component embeds test descriptions, test specifications, pass/fail criteria, etc. In general terms, this component is necessary to ensure that the overall test modalities and parameters are predefined and provides input for further components of the entire OTA.

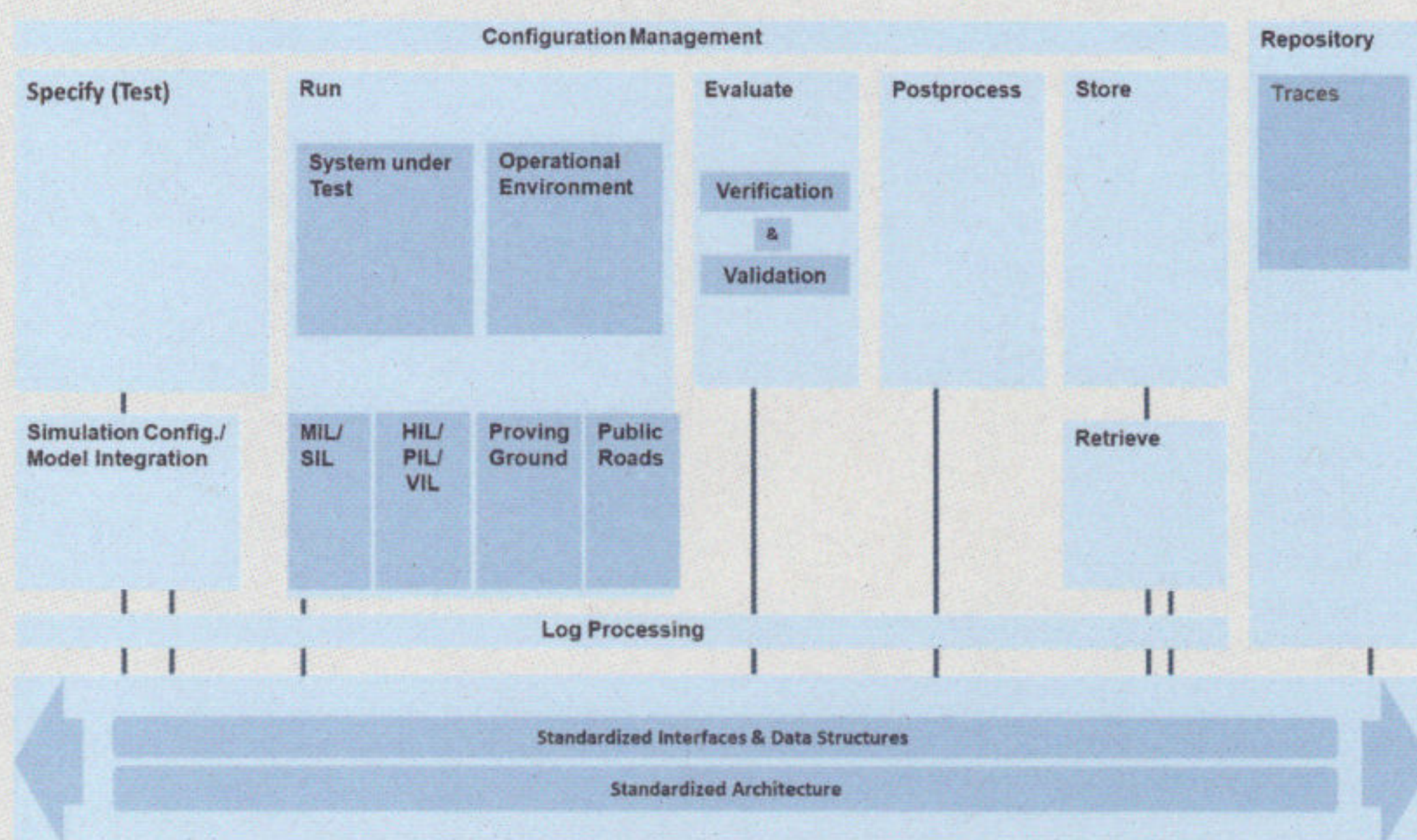


Figure 1.6: Generic open testing architecture (OTA) [4]

The **simulation configuration/model integration** component integrates the pre-configuration of the run component. Setting up the simulation-run environment requires basic simulation configurations, such as simulation time management, required observers, choice of numerical solvers, etc., model integration, model coupling and parameter selection. It can be beneficial to utilize more than one simulation tool to take advantage of a variety of specialized simulation tools simultaneously. For example, the advantage of using a vehicle dynamics and traffic simulation as a co-simulation has already been demonstrated in [6], [7] and [8]. Both the effects and artefacts of model and tool coupling need particularly intensive investigation and diligence because the numerical errors and logical fallacies are not always trivial and easy to identify. Additionally, the hardware setup needs to be configured within this component. This applies to solely virtual MiL/SiL setups as well as to Hardware-in-the-Loop (HiL)/Prototype-in-the-Loop (PiL)/Vehicle-in-the-Loop (ViL) hybrid reality methods and even real-world testing on proving grounds or public roads.

The core of the entire OTA is the **run** component, which compiles and processes the tests. For this reason, the run component requires several sub-components within itself in order to operate properly. First, the system under test (SUT), also known as the CAV, or a sub-component such as an AI-based perception module, for example, are incorporated. Furthermore, the operational environment within which the SUT is supposed to operate is the second sub-component within the run component.

These sub-components are necessary for all testing methods and deployment procedures. The configuration of the SUT with respect to the operational environment itself is especially important. When specifying a test, the question of the context or overall system in which the SUT needs to be tested arises. Thus, the SUT can vary from a single component or entire system, such as a CAV, up to a traffic system or entire city. Likewise, the operational environment in which the SUT is tested can be described as anything from a single component to an entire traffic system. The OTA allows extensive freedom in setting up the SUT and operational environment, subject to the proviso that both need to be specified for the run component to operate. Additionally, the hardware for the necessary calculations needs to be factored into and specified in this component. For virtual

methods, such as MiL/SiL the hardware setup accounts for computational power and maybe a distribution over several specialized computation clusters. In terms of test methods that are no longer solely virtual, the OTA provides the option of incorporating HiL/PiL/ViL approaches up to proving ground and public road testing. Of course, the considerations for these approaches in terms of computing power and hardware differ from purely virtual testing. Nevertheless, hardware design is an important part of the run component and needs to be carried out with diligence.

Test evaluation and analysis takes place in the **evaluate** component, the purpose of which is self-explanatory. Metrics, functional and non-functional requirements as well as pass/fail criteria are calculated within this component. Additionally, the component can estimate the accuracy of models and solvers as well as cumulative failures during the simulation run. Thus, the component provides vital information on overall test integrity by performing an accuracy and validity assessment.

In order to reduce computation time, the **post-process** component makes it possible to calculate deduced test criteria and metrics on the basis of a variety of measured signals. While this step could also be conducted in the evaluation component, the more complex it becomes to calculate certain metrics, the greater the preferability of doing so once the test has been completed.

The **repository** provides the an option for permanently storing such things as specific or overall OTA configurations, models, data, calculated test results and setups. As such, it serves as an overall storage unit for the entire OTA.

Repository access and data storage are managed by the **Store** and **Retrieve** components, respectively. These components can also process complex queries.

Every bus system requires simple data logging function that is connected to all the other components in the system. For this reason, a **log processing** component is embedded in the OTA.

An overview of relevant standards for the OTA and a more compact description can be found in the ASAM Guide: Simulation [4].

1.3.2 Automated configuration/set-up

It is important to note that the configuration and set-up of the OTA and, consequently, every other future testing procedure cannot be done manually, simply because of the huge effort involved in CAV testing in general. Automated configuration can help a lot by standardizing the entire OTA, interfaces, data structures and overall component configuration. While this may be a challenge and additional source of work at the outset, it will become a key enabler in the long run. Recent CAV testing initiatives have shown repeatedly that using short cuts to configure and set up testing procedures in the first place is a game changer when the long-term overall release of CAV safety is seriously pursued. Thus, the additional time and investment that need to be spent at the outset will prove to be of considerable value in the long run.

The issue of what exactly automated configuration/setup means remains a little blurry along the edges because of the many nuances in testing procedures, architectures, components, etc. The authors would now like to illustrate some aspects of this topic as a way of providing suggestions for setting up future testing architectures and procedures. There needs to be a formal and standardized way of setting up components and their interfaces to permit automated configuration. The basic structure should be predetermined, while the interfaces and data structures for communication with other components are already specified by the OTA bus system. The incorporation of code, simulation models and parameters can be defined by the developer himself within the basic framework provided by the predetermined structure. This makes it possible to utilize a variety of simulation models, parameter configurations, etc. that are tailored to developers' demands while still guaranteeing automated configuration/setup that is compatible as "simulation as a service".

1.3.3 Simulation as a service (SAAS)

The SAAS supported and enabled by the OTA affords the option of utilizing components as well as tools across domains that are supplier-neutral. It is thus possible to combine a multitude of commercially available simulation tools, models, evaluation metrics and test specifications from different suppliers in such a way as to take advantage of free market principals. The days where suppliers offered one end-to-end solution for all problems are definitely over. Nowadays, different solutions have

to collaborate and the component exchangeability is needed, which manifests itself in a dominant mode of thought and a generally preferred engineering design principal for CAV testing.

All in all, SAAS will shape future development, test and validation procedures for both AI components and CAVs. Current engineering design principals, which are already outdated for this engineering task, will be gradually replaced by "as a service" approaches, and these will shape the future of complex testing procedures.

1.4 Engineering tasks

The proposed OTA allows for a diverse range of different engineering tasks in the field of CAV and AI testing. To keep this section sufficiently brief, the authors will focus on the following example engineering tasks, while noting that these do not indicate any restriction of applicability. Thus, the OTA supports the entire spectrum of CAV and AI testing.

The engineering task of identifying critical scenarios [6], which can be regarded as a sub-set of scenario mining, is a well investigated task that relies quite heavily on simulation. The main challenge of these approaches is to mine scenarios in the light of a predefined metric that reflects requirements that determine which scenarios should be identified. Criticality parameters are primarily used to mine challenging scenarios that are subsequently used in testing in the validation phase of a CAV or AI component. The scenario mining schemes are usually set up in a unique fashion based on a commercial simulation tool. While this way of development is obviously outdated, the issue can be resolved using the OTA. The advantage of using various components, models and tools in a standardized architecture is that it maximizes independence and exchangeability. Even the metrics that are of vital importance for these methods can be exchanged easily. The same applies to tools and models, as well as to parameter sets, which change frequently during testing.

Furthermore, the search for "corner cases" and their safety-related role is another important engineering task that is considered here. Work on investigating corner cases focuses on the limits of the known parameter spaces, on the corners of extreme circumstances that do not often occur. The purpose of the tests that are performed is to generate knowledge about the extreme limits of the parameter spaces. This is another area where

the flexibility of the OTA can be exploited. In this case, the exchangeability of models is important, because the validity of the simulation models is deliberately directed to the edges of parameter spaces in order to permit corner case testing.

Direct access to the repository of critical scenarios provides a starting point for further investigation. The OTA makes it possible to exchange, use and reuse critical scenarios for different test environments, such as MIL, SIL, HIL, etc. Thus, the OTA provides a flexible way of incorporating these test environments and utilizing them across various engineering tasks. In the case of corner cases, combining the OTA with a testing procedure can direct testing activities towards a predefined metric, for example, criticality indicators such as TTC. If required, the test environment can be exchanged to improve test validity by using HiL/PiL or even tests conducted on a proving ground up to data gathered on public roads.

This exemplifies how the OTA supports engineering tasks and deployment procedures and offers a novel approach to using automatically configured SAAS schemes to test AI components and CAVs.

1.5 Simulation-enabled testing and deployment procedures

Considering the entire spectrum of testing and deployment procedures that simulation enables, ranging from purely virtual procedures to tests on public roads, a number of important characteristics and differentiating factors need to be addressed. It is not necessarily the case that a test on a proving ground is more valid than a simulation or vice versa. Depending on the measurement equipment available at a proving ground, inaccuracies arise. Furthermore, it is often claimed that simulation is cheaper and more efficient than real-world testing. This is also debatable, given the computing power required to model the entire world accurately in a simulation run. However, the claim that virtual testing requires less effort holds to some degree. Its validity rises when the procedure tends towards real-world testing by design using, for example, HiL/PiL/ViL. Conversely, this does not mean aiming for perfection when using virtual methods, but introducing more real-world elements into the testing procedures rather than constantly increasing the quality and complexity of the simulation models. This is in line with the idea that simulation results should be valid enough for the engineering task in question.

If greater quality and precision are needed, mixed and hybrid-reality procedures come into play. These procedures use real components and confront them with virtual stimuli. For example, a real-world CAV is driving on a proving ground while the other test participants are simulated. The sensor object-list can be overwritten to deceive the CAV into assuming that other participants are performing driving manoeuvres nearby, and the CAV reacts accordingly. This is an alternative to proving ground tests utilizing other test vehicles and possesses many advantages, including test reproducibility, damage mitigation when testing critical scenarios, etc.

Introducing real-world test vehicles on a proving ground can be considered the next step towards real-world testing. Nevertheless, the fact that the tests are performed in a confined space constitutes a simplification even though all of the test participants – including the test environment – are real-world objects. The test results are error-prone due to measurement inaccuracies. Even if an independent monitoring system is installed, measurement precision increases but cannot be mitigated entirely.

The same can be argued for tests conducted on public roads. In this case, there is no longer any simplification compared to virtual and hybrid reality procedures or proving ground tests. The real bottleneck for public road testing is the monitoring system, which relies on a variety of sensor perceptions, making the measurement inaccuracies a difficult challenge to overcome. Monitoring systems rely on various sensors simultaneously to increase measurement integrity and minimize errors, such as misclassifications.

As a framework, the OTA's run component provides all the above-mentioned testing and deployment procedures summarized for virtual testing SIL/MIL, mixed and hybrid reality approaches HiL/PiL/ViL as well as proving ground and public road testing.

1.5.1 Parameter variation

Parameter variation is vital for comprehensive testing. For this reason, the OTA predefines the SUT and operational environment as key aspects within the run component. It is important to notice that the parameter variation described in this section is intended for test design and thus not for numerical precision nor the non-design parameters that are embedded in the simulation models

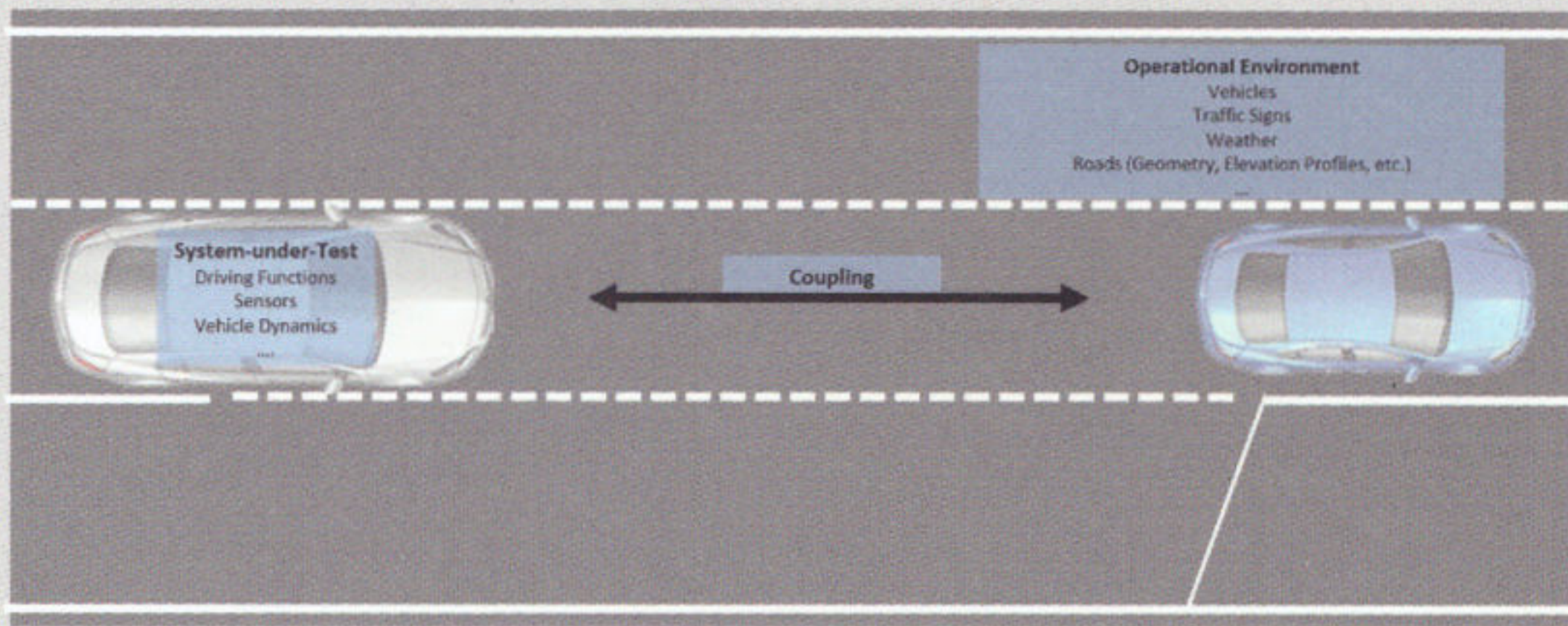


Figure 1.7: Parameters of a CAV (system under test, SUT) driving on a highway (operational environment)

for validity reasons. The parameters described are therefore designed to create a variety of circumstances which create systematic tests for CAVs and AI components. Figure 1.7 shows the SUT and the operational environment, including parameterizable examples.

In this example, the SUT is defined as a CAV driving on a highway representing the operational environment. Note that this configuration can be entirely different, as mentioned before, but this example is being used in this case to explain the aspects of this section. The CAV in this example consists of driving functions, sensors and vehicle dynamics. This represents the minimum set needed to model a CAV. The operational environment contains static components, such as roads, road markings and traffic signs, as well as dynamic components, such as vehicles and weather conditions, to name but a few. All of these components can be parameterized within a certain bandwidth. The parameters can be static, a part of a model and therefore described as a differential equation or as an automaton.

Tests can be generated through specific variation of these parameters. Now the basic principle of parameter variation has been clarified, the specifics and details for these examples can be discussed. In addition, questions arise if the non-design parameters are varied as well and what kind of effects are anticipated and if the design parameters

can be decoupled from non-design parameters in all cases.

To keep this section brief, further examples and engineering tasks will not be added.

1.5.2 Cooperative and automated vehicle (CAV)

Given the parameters of a CAV, it seems reasonable to separate the design parameters from the non-design parameters. It should be noted that the non-design parameters, such as vehicle mass and center of gravity, can be very well design parameters in the general vehicle development process, but not when focusing on testing of CAV driving functions. These so-called non design parameters play a vital role in determining simulation model quality and need to be generally considered while designing a motion control algorithm, for example. As regards test generation, the more interesting design parameters are those used to make driving strategy decisions, fuse the sensor perceptions or control vehicle motion. Figure 1.8 shows a CAV with example design and non-design parameters.

CAV design parameters exist along the sense-plan-act robotic paradigm, an established paradigm in CAV development. Parameters can be varied in a multitude of dimensions throughout the paradigm. The challenge to be overcome emerges in the development of methodologies on how to vary parameters purposefully and effectively. The

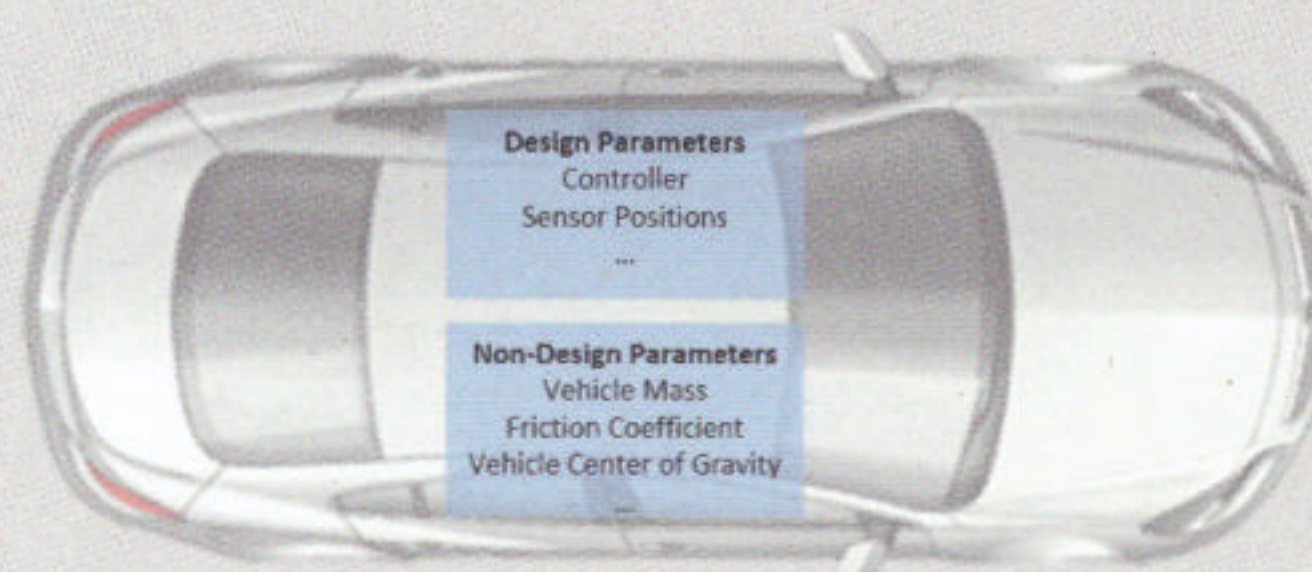


Figure 1.8: Example design and non-design parameters of a CAV

controller parameters, for example, directly influence the motion of the CAV, resulting in a more aggressive or defensive elimination of the control difference based on the reference value provided by the trajectory planner. Changing these parameters has a direct effect on safety while simultaneously influencing other requirements, such as energy efficiency, driving comfort, etc. Joint efforts to resolve these issues can be found in the PEGASUS family initiative [9], [10], [11] and published research papers [12], [13], [14] and [15], to name but a few sources for the interested reader.

At all events, the OTA provides the option of incorporating yet undeveloped methodologies in the illustrated components. The OTA's bus system-like configuration makes it possible to enhance further OTA components that might be required once these methodologies have been developed and agreed.

1.5.3 Operational environment

Parameters for test generation are mostly dependent on the domain the SUT is supposed to operate in, also known as the operational environment. It is not a trivial task to describe the operational environment systematically in all its complexity. Researchers within the PEGASUS family initiative designed a six-layer model categorizing a vertical layer structure comprising layer 1 (road networks), layer 2 (roadside structures), layer 3 (temporary modifications of L1 and L2), layer 4 (dynamic objects), layer 5 (environmental conditions), layer 6 (digital information) [16]. Within this structure, parameters can be assigned and varied within a predefined structure and categorization. Considering the OTA, this model can be used as a guideline for setting up the operational environment and clustering parameters.

The 6-layer model currently addresses AI-specific features that could be added to enhance the operational environment description. The AI-specific features mostly focus on perception, and for this reason, the model should be enhanced by adding sensor-specific aspects as well as reflecting object properties that affect sensor impressions. Sensor

characteristics are difficult to model using simulation approaches while, unfortunately, perception failures are one of the major causes of critical scenarios. Since perception is the first part of the sense-plan-act paradigm, errors made at the start of the chain are difficult to eliminate later on.

An additional challenge, especially for simulation-based approaches, is going to be the validation of the digital twin of the operational environment. A lot of effort is going into developing the perfect sensor models for standard camera, radar and lidar sensors, but the digital world and all the reflections of the objects in the environment must be modelled as well in order to obtain an accurate sensor impression. This task is even more difficult than sensor modelling itself and needs further investigation in subsequent research.

However, the issues of how significant the criticality parameters are and how the deduction of critical scenarios as well as criticality parameter regions can be identified in an efficient and practical manner are still subject of current research. Most current approaches lack practicability, or to put it in more directly terms, current proposed methodologies still have to withstand the credibility test best summed up by saying the tyre needs to hit the road at some point. In other words, theoretical considerations need to be applicable and practicable.

Whatever methodology, or, as is more likely, set of methodologies prevail, the OTA can provide a framework consisting of every necessary simulation-enabled testing and deployment procedure. Thus, there are no enforced restrictions on methodologies and approaches. Because of the generic setup, the OTA supports the application of all methodologies that are developed for future testing.

1.6 Summary and outlook

The application landscape for simulation-enabled development, test and validation methods appears to be continuing to grow. It is noticeable that this growth seems to be accelerating over time, indicating that demand for these methodologies is more likely to increase in the future. That is why these methods require sophisticated frameworks and architectures to handle the complexity that in-

trinsically arises when focusing on the development, testing and validation of applications, such as cooperative and automated driving and AI components.

In recent years the DevOps cycle has gained increasing popularity among the engineering community working on the above-mentioned applications. This can be attributed to this framework's advantage in covering the entire product lifecycle. Moreover, it makes updates and functional improvements to systems that have been released and are in use feasible.

Within this framework, simulation needs a generic architecture that permits exchangeability and standardized components and interfaces. The OTA provides this in a bus system-like setup. This architecture allows for SAAS procedures while permitting automated configuration of both individual components and the entire architecture.

The possibility of tackling a large variety of engineering tasks, such as identifying critical scenarios, as well as simulation-enabled testing and deployment procedures such as systematic parameter variations, gives the OTA a major advantage when combined with existing simulation-based methods and procedures.

The aspects presented in this article are key areas for future research and refinement. Further investigations of applicability to various engineering tasks in practice need to be carried out. It is going to be interesting to see which engineering tasks will be challenged utilizing these procedures, methods and tools.

The authors conclude by emphasizing the importance of standardization in this field, since a high level of standardization is crucial to satisfying the requirements outlined in this article.

Bibliography

- [1] F. Koester and S. Rude, SET Level Mid-term Event – Introduction and Challenges, Munich: SETLevel Mid-term Event, 2021.
- [2] U. Eberle, S. Hallerbach, R. Mannale, B. Kramer, C. Neurohr, M. Steimle and C. Amersbach, The Pegasus Method for Safeguarding Automated Driving: What else is needed?, Wolfsburg: Pegasus Final Event and Symposium, 2019.
- [3] V. Lizenberg, B. Buechs, S. Knapp, R. Mannale and F. Koester, Graphical Data Visualization for Vehicular Communication Systems in Real and Virtual Test Environments, Dortmund: AmE (Automotive meets Electronics), 2020.
- [4] F. Koester, S. Hallerbach, P. Mai and B. Engel, "ASAM OpenX in Context," *ASAM SIM:Guide – Standardization for Highly Automated Driving*, pp. 38-49, 2021.
- [5] F. Koester, S. Rude and S. Hallerbach, Methodologies for Simulation-Based Engineering Tasks – Towards a SET Level Methodology, Munich: SETLevel Mid-term Event, 2021.
- [6] S. Hallerbach, Dissertation: Simulation-Based Testing of Cooperative and Automated Vehicles, Oldenburg, 2020.
- [7] S. Hallerbach, Y. Xia, U. Eberle and F. Koester, "Simulation-Based Identification of Critical Scenarios for Cooperative and Automated Vehicles," *SAE International Journal of Connected and Automated Vehicles* 1(2), p. 93–106, 2018.
- [8] S. Hallerbach and U. Eberle, "Simulation-based Methods for Development, Test and Validation of Automated Driving: Scenarios, Smart Agents and Hybrid Reality," SATW Workshop "Connected and Automated Driving: In-vehicle and Deployment, Swiss Academy of Sciences, Zurich, 2020.
- [9] PEGASUS-Family, "PEGASUS RESEARCH PROJECT – SECURING AUTOMATED DRIVING EFFECTIVELY," [Online]. Available: <https://www.pegasusprojekt.de/>. [Accessed 13/08/2021].
- [10] PEGASUS-Family, "Verification and validation methods for automated vehicles in urban environments – Safety standards for automated driving," [Online]. Available: <https://www.vvm-projekt.de/>. [Accessed 13/08/2021].
- [11] PEGASUS-Family, "SETLevel – Simulation-based Development and Testing of Automated Driving," [Online]. Available: <https://setlevel.de/>. [Accessed 13/08/2021].
- [12] C. Neurohr, L. Westhofen, H. Tabea, G. d. Thies, E. Moehlmann and E. Boede, "Fundamental Considerations around Scenario-Based Testing for Automated Driving," *IEEE Intelligent Vehicles Symposium (IV)*, pp. 121-127, 2020.
- [13] V. Lizenberg, D. Bischoff, Y. Haridy, U. Eberle, S. Knapp and F. Koester, "Simulation-Based Evaluation of Cooperative Maneuver Coordination and Its Impact on Traffic Quality," *SAE Technical Papers (01):0171*, 2021.
- [14] B. Schuett, M. Steimle, B. Kramer, D. Behnecke and E. Sax, "A taxonomy for quality in simulation-based development and testing of automated driving systems," Preprint, 2021.
- [15] N. Weber, D. Frerichs and U. Eberle, A simulation-based, statistical approach for the derivation of concrete scenarios for the release of highly automated driving functions, Dortmund, AmE (Automotive meets Electronics), 2020.
- [16] M. Scholtes, L. Westhofen, L. Turner, K. Lotto, M. Schuldes, H. Weber, N. Wagener, C. Neurohr, M. Bollmann, F. Koertke, J. Hiller, M. Hoss, J. Bock and L. Eckstein, "6-Layer Model for a StructuredDescription and Categorization of UrbanTraffic and Environment," *IEEE Access*, vol. 9, pp. 59131-59147, 2021.

Citation Information

S. Hallerbach, U. Eberle, F. Köster, „Simulation-Enabled Methods for the Development, Testing and Validation of Cooperative and Automated vehicles” in SATW-Fokus-Dokument “Autonomes Fahren. Treiber zukünftiger Mobilität“, p. 30-41, Schweizerische Akademie der technischen Wissenschaften (satw), February 2022.

References

- [1] F. Koester and S. Rude, SET Level Mid-term Event - Introduction and Challenges, Munich: SETLevel Mid-term Event, 2021.
- [2] U. Eberle, S. Hallerbach, R. Mannale, B. Kramer , C. Neurohr, M. Steimle and C. Amersbach, The Pegasus Method for Safeguarding Automated Driving: What else is needed?, Wolfsburg: Pegasus Final Event and Symposium, 2019.
- [3] V. Lizenberg, B. Buechs, S. Knapp, R. Mannale and F. Koester , Graphical Data Visualization for Vehicular Communication Systems in Real and Virtual Test Environments, Dortmund: AmE (Automotive meets Electronics), 2020.
- [4] F. Koester, S. Hallerbach, P. Mai and B. Engel , "ASAM OpenX in Context," *ASAM SIM:Guide - Standardization for Highly Automated Driving*, pp. 38-49, 2021.
- [5] F. Koester, S. Rude and S. Hallerbach, Methodologies for Simulation-Based Engineering Tasks - Towards a SET Level Methodology, Munich: SETLevel Mid-term Event, 2021.
- [6] S. Hallerbach, Dissertation: Simulation-Based Testing of Cooperative and Automated Vehicles, Oldenburg, 2020.
- [7] S. Hallerbach, Y. Xia, U. Eberle and F. Koester, "Simulation-Based Identification of Critical Scenarios for Cooperative and Automated Vehicles," *SAE International Journal of Connected and Automated Vehicles* 1(2), p. 93–106, 2018.

- [8] S. Hallerbach and U. Eberle , "Simulation-based Methods for Development, Test and Validation of Automated Driving: Scenarios, Smart Agents and Hybrid Reality," SATW Workshop "Connected and Automated Driving: In-vehicle and Deployment, Swiss Academy of Sciences, Zurich, 2020.
- [9] PEGASUS-Family, "PEGASUS RESEARCH PROJECT - SECURING AUTOMATED DRIVING EFFECTIVELY," [Online]. Available: <https://www.pegasusprojekt.de/>. [Accessed 13 08 2021].
- [10] PEGASUS-Family, "Verification and validation methods for automated vehicles in urban environments - Safety standards for automated driving," [Online]. Available: <https://www.vvm-projekt.de/>. [Accessed 13 08 2021].
- [11] PEGASUS-Family, "SETLevel - Simulation-based Development and Testing of Automated Driving," [Online]. Available: <https://setlevel.de/>. [Accessed 13 08 2021].
- [12] C. Neurohr, L. Westhofen, H. Tabea, G. d. Thies, E. Moehlmann and E. Boede, "Fundamental Considerations around Scenario-Based Testing for Automated Driving," *IEEE Intelligent Vehicles Symposium (IV)*, pp. 121-127, 2020.
- [13] V. Lizenberg, D. Bischoff, Y. Haridy, U. Eberle, S. Knapp and F. Koester, "Simulation-Based Evaluation of Cooperative Maneuver Coordination and Its Impact on Traffic Quality," *SAE Technical Papers (01):0171*, 2021.
- [14] B. Schuett, M. Steimle, B. Kramer, D. Behnecke and E. Sax, "A taxonomy for quality in simulation-based development and testing of automated driving systems," *Preprint*, 2021.
- [15] N. Weber, D. Frerichs and U. Eberle, "A simulation-based, statistical approach for the derivation of concrete scenarios for the release of highly automated driving functions," *AmE 2020 - Automotive meets Electronics; 11th GMM-Symposium*, 2020, pp. 1-6.
- [16] M. Scholtes, L. Westhofen, L. Turner, K. Lotto, M. Schuldes, H. Weber, N. Wagener, C. Neurohr, M. Bollmann, F. Koertke, J. Hiller, M. Hoss, J. Bock and L. Eckstein, "6-Layer Model for a StructuredDescription and Categorization of UrbanTraffic and Environment," *IEEE Access*, vol. 9, pp. 59131-59147, 2021.