Seamless Tool Chain for the Verification, Validation and Homologation of Automated Driving

Andrea Leitner, Jürgen Holzinger, Hannes Schneider, Michael Paulweber, and Nadja Marko

1 Introduction

Autonomous vehicles are becoming increasingly capable and frequent. Any automated application, whether Advanced Driver Assistance Systems (ADAS) or Highly Automated Driving (HAD) functions, needs to go through rigorous testing procedures due to the traditionally strict safety requirements of the automotive industry. However, purely physical testing is cost- and time consuming. Varying road infrastructure, changing traffic and weather conditions, different driver behavior and country-specifics need be considered. Furthermore, there is a heavy tail distribution of surprises from the real world and an automated vehicle needs to be able to detect and react to these unforeseen situations [1]. All this makes a so-called proven-in-use certification solely based on physical test driving almost impossible [2]. Software-based (virtual) validation approaches are therefore promoted as a viable alternative to support the testing process by enabling a higher scenario coverage at lower costs and in shorter time. Simulations can further be used to identify the sub-set of test cases that should be executed in a more accurate test environment (Vehicle-in-the-Loop or proving ground). At the end, the whole validation process will be a mixture of different test environments and the results of this overall process will be used as an input for homologation. Especially for virtual test environments it is also important to show the correctness and accuracy of the simulation results. This means that also all models used in the virtual environment must be compared to real world behavior to ensure trustable results [3]. For all these different tasks it is important

A. Leitner (✉) · J. Holzinger · H. Schneider · M. Paulweber
AVL List GmbH, Graz, Austria
e-mail: andrea.leitner@avl.com

N. Marko
Virtual Vehicle Research Center, Graz, Austria

A. Leitner et al. (eds.), *Validation and Verification of Automated Systems*,
https://doi.org/10.1007/978-3-030-14628-3_14

to have a seamless toolchain, where models as well as test cases can be reused independent of the test environment.

This paper discusses the requirements and potential solutions for the realization of a seamless toolchain for the validation of automated driving functions.

2 Test Execution Platforms in the Context of the Generic Test Architecture

Figure 1 shows an exemplary instantiation of the generic test architecture [15] proposed by the ENABLE-S3 project and introduced in chapter "ENABLE-S3: Project Introduction". The focus of this paper is on the lower layer called Test Execution Platform. This layer subsumes the different test environments from purely virtual testing to real-world testing.

In a seamless toolchain, we do not only assume that models can be reused throughout the different test environments, but also that there is a common interface for test cases as well as for collecting test results. This enables the reuse of test cases and test plans on different test execution environments. Of course, the Test Planning tool has to take care about preparing the test cases in a way that they are executable in a specific test environment (i.e. considering the constraints and formats of the execution environment).

One dimension in Fig. 1 distinguishes between three phases: *Preparation*, *Execution* and *Evaluation & Reporting*. The focus of this paper is on the Execution phase, but for completeness, we provide a brief introduction into the other phases as well.

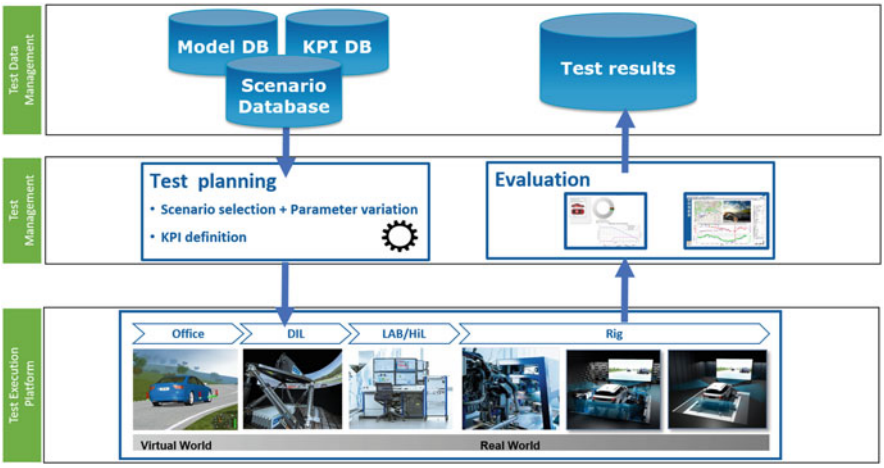

Fig. 1 Exemplary instantiation of the generic test architecture with different Test Execution Platforms

The *Preparation* phase includes all tasks, which are required for preparing the test execution. This basically means that the test input data needs to be selected and prepared in a way that makes it executable. An important asset for scenario-based validation are of course scenarios. Scenarios can come from different sources (e.g. engineered or extracted from measurement data) and consists of different aspects, such as dynamic aspects (traffic participants), static aspects (road and traffic signs) and the scenery (3D environment as input for sensor models). All these aspects need to be represented in a scenario database. For test case generation, these aspects are combined and parametrized in a certain way. Of course, the level of detail and the format is dependent on the test environment. For virtual environments, OpenDrive[1] and OpenScenario[2] can be used as standardized formats. For proving ground testing, the scenarios need to be transformed in a format, which can be used for test equipment automation. In addition, KPIs need to be assigned to the test cases dependent on the test purpose to configure the required measurements. As a last step, the test execution platform configuration needs to be prepared. This includes the selection of the required models and hardware components and their communication. In general, there are two ways of test case generation. In offline test case generation, a complete test plan is prepared beforehand. This test plan consists of a series of test cases, which can be executed either in a virtual test environment or on the proving ground (see e.g. [4]). For online test case generation, an initial test plan is prepared and based on the results of these test cases further test cases are generated and executed on-the-fly (see e.g. [5]).

The *Evaluation & Reporting* phase takes the results of the test execution and assesses them. This can again be done online or offline. In the case, where test plans are updated on-the-fly, the respective KPIs also need to be calculated during the execution or immediately afterwards. Especially on the proving ground, measurements need to be taken and uploaded for further processing. Either way, we propose that the same data format is used in order to be able to reuse the KPI evaluation scripts. In our case, the Open Simulation Interface (OSI) [6] specification is used to describe sensor measurement data, no matter if it is coming from simulation or from real sensors.

The *Execution* phase includes the test execution environments. They and their potential usages are described in more detail below.

Cloud/HPC Simulation Especially in early phase testing, the huge test space needs to be scanned quickly. This requires a scalable and fast simulation environment, where many different parameter combinations can be tested in parallel. Usually, simulation environments are built for desktop simulation and are not prepared for scalability. This is especially true for co-simulations, which include various simulation tools. To support fast and scalable simulation, different cloud-based

[1] https://www.asam.net/standards/detail/opendrive/

[2] https://www.asam.net/standards/detail/openscenario/

simulation environments have popped-up recently (e.g. Metamoto,[3] Cognata,[4] etc.). These companies promise Simulation as a Service to train, test, debug, and validate automated vehicle software. Model.CONNECT provides a solution based on Docker Containers,[5] which can be used for setting up scalable co-simulations. The main advantage is the ability to use the same simulation infrastructure and models as for other test environments. Additionally, the interfaces to test case generation and test evaluation tools stay the same and can be used in the same way as for other test environments.

This setup can also be used for continuous integration and running test cases automatically for each new software version.

Model (Software) in the Loop This non-real-time test environment features all system aspects as models or software components. It is mainly used by function developers, who are testing their function (sensor fusion, trajectory planning, etc.) or reproducing erroneous situations identified in real-world testing. It further allows to compare different sensor concepts and algorithm parameterizations in a safe und reproducible way even before the availability of actual hardware prototypes. There are many providers for simulation environments (e.g. Vires VTD,[6] TASS PreScan,[7] Oktal Scaner,[8] etc.). All have their strength and weaknesses and depending on the testing purpose the one or the other might be suited better. There are also several in-house solutions to meet the specific requirements of companies. With respect to the overall toolchain, an important requirement of a simulation environment is the possibility to integrate it with real-time components to be able to reuse the same environment also in combination with hardware. This requires a modular structure and certain interface capabilities.

Hardware in the Loop Hardware in the loop is a very common environment to test different kinds of control units standalone or in combination. The main purpose is to identify timing and communication issues.

Driver (Human) in the Loop Driver/human-in-the-loop simulation provides a highly realistic driving experience to the human/driver which is indispensable for investigating the influence of new products on human [7].

Vehicle in the Loop This test environment bridges the gap between conventional HiL testing and real road testing for efficient and reproducible validation of fully integrated vehicles. One concept for vehicle in the loop testing is the DrivingCube concept [8]. The advantage is the transfer of tests to a controlled environment where

[3] https://www.metamoto.com
[4] https://www.cognata.com
[5] https://www.docker.com
[6] https://vires.com/vtd-vires-virtual-test-drive
[7] https://tass.plm.automation.siemens.com/prescan
[8] https://www.avsimulation.fr

the risk for man and machine is reduced and the effectiveness is increased. It can be either a chassis dynamometer or a powertrain test bed for complete vehicles. Since the interaction with the steering system was not considered until now, the test of lateral controllers was not feasible. Nevertheless, new developments enable the stimulation of the steering system of the vehicle while keeping the interference with the vehicle at a minimum. This is achieved by a mechanical decoupling of tires and the steering system. Instead, a compact and universal steering force module is used to induce forces to the tie rod [9]. An additional requirement is the need to stimulate the environment sensors. There are several approaches for sensor stimulation (e.g. over-the-air stimulation of radar sensors [10] or the use of moving bases [11]).

Proving Ground Currently, tests on proving grounds are mainly focusing on standardized testing of active safety systems [12] and evolving gradually towards driver assistant systems. These testing approaches will not scale for automated driving, which means that test automation and new techniques (e.g. augmented reality on proving grounds) are required. One challenge is the increased complexity of orchestrating a growing number of robotic test equipment on proving grounds. Another one is the ability to collect, exchange, and compare data from test tracks as well as from virtual test environments. This requires a certain standardized data format [13]. Currently, for testing of active safety systems, the proving ground is the last instance of testing activities. Nevertheless, with the growing importance of virtual validation, proving grounds will also be an important environment for model calibration and validation.

Road testing is of course still a required test environment. Nevertheless, because of the different scope and requirements (no test planning, no models, etc.) we will not discuss this test environment in detail here.

In our setup, the co-simulation platform Model.CONNECT[9] and real-time integration platform Testbed.CONNECT[10] are used to couple the different elements (models and hardware components) and to take care about the communication and timing issues between these elements. Model.CONNECT interlinks simulation models into a consistent virtual prototype, regardless of the tool they were created with. Simulation and hardware components can be easily integrated into a complete virtual/real system. This facilitates the continuous, model-based development in a wide range of powertrain and vehicle applications (e.g. driver assistance systems). Testbed.CONNECT connects simulation models with the testbed. The testbed engineers do not have to wait for all the hardware components to be available but can simply replace them with the corresponding simulation models. Even complex models from the concept phases can be easily and robustly integrated on any kind of testbed. The system developers gain a deeper understanding of the complex interactions of their systems by using their models at the testbed. The acquired findings are continuously used to the further improve the simulation models. The

[9] https://www.avl.com/-/model-connect-

[10] https://www.avl.com/web/guest/-/testbed-connect-

concrete architecture used within the co-simulation and integration platform is described in more detail in Sect. 3.

Model.CONNECT is furthermore used as the interface for test planning and evaluation. This means that Model.CONNECT is taking care about the execution of the test cases in the respective test environment. There is also one interface to collect the simulation results for evaluation and post-processing.

Another important building block is a model repository, which can be used to manage and access different versions of different simulation models (e.g. various levels of detail for different testing purposes). The tight integration in Model.CONNECT enables the definition of different simulation configurations, which can be switched on the fly as needed.

3 Test Execution Environment: Architecture

The Test Execution Platform covers all relevant aspects of an automated cyber physical system as described in chapter "ENABLE-S3: Project Introduction" and shown in Fig. 2. An essential aspect of automated driving functions is their tight interaction with the environment (i.e. other traffic participants). This means that the environment needs to be simulated in an appropriate manner. For the perception of the environment, the automated vehicle uses different kinds of sensors. Therefore, it is also important to represent the characteristics and potential limitations of the sensors in terms of sensor models in the simulation. The required level of detail and accuracy of these models depend on the testing purpose. For testing the trajectory planning, ideal object level sensor models might be sufficient. For testing sensor fusion algorithms or even for system validation, more accurate and detailed sensor models (either phenomenological or physical) reflecting the properties of the actual sensor types are required. If the automated vehicle uses communication with the infrastructure or other vehicles, this aspect needs to be considered in simulation as well.

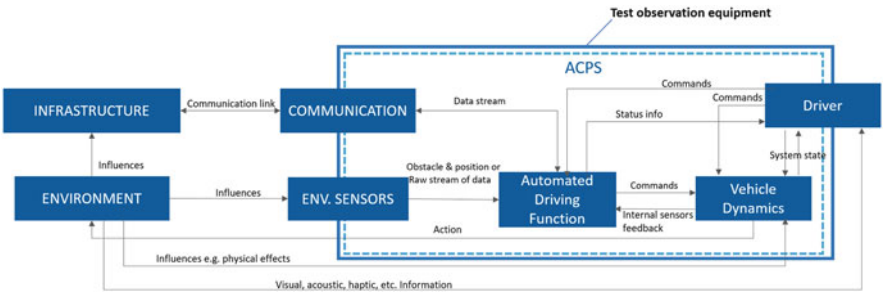

Fig. 2 Test execution platform—detailed architecture

Depending on the function under test, the dynamics of the vehicle need to be modeled as well. It gets evident that a modular structure is important here as well, since the required level of detail of the models differs for different tests. A modular structure enables an efficient exchange of models. Usually these models are developed using different simulation tools, each specialized on a specific aspect.

A co-simulation platform is required to couple the different simulation models to a holistic closed-loop simulation. The co-simulation platform is responsible for establishing the communication and minimizing latency effects during simulation. Depending on the development stage, there will be different instances of the test execution platform. For example, in a MiL environment all components will be available as simulation models. Later simulated components will be step by step substituted by real physical components resulting in a mixed environment of real-time and non-real-time components. Especially this latter case presents additional challenges for the test execution platform as there are various real-time systems with different properties that have to be integrated. Additional requirements have to be fulfilled; there are hard real-time conditions that have to be considered, communication is needed in real-time, synchronization of real-time and wall clock time as well as communication delays have to be handled by the platform. Real hardware has to be operated safely which means that safety mechanisms have to be implemented in order to avoid damage of hardware. To facilitate the integration of different models and real-time components, standardized interfaces can be used (FMI[11] for non-real-time communication and DCP[12] for real-time communication).

Another important aspect, which influences the performance of the simulation is the system decomposition. The main questions are how to split aspects in different models, how to distribute the execution of the models and how to define the model interfaces. Splitting and distributing models can be an important mean to improve the performance (e.g. distribution to different cores). The definition of model interfaces is often constraint by I/O capabilities of simulation tools but can also include a lot of integration experience and know how (e.g. taking constraints by physics as low inertia vs. high inertia into consideration). This knowledge could be made explicit and reusable by providing best practice templates and standardized interfaces.

4 Open Simulation Interface

Traditional automotive testing is mainly based on time-series signals and most tools are designed and optimized for this kind of data. For testing automated cyber-physical systems, large data sets with complex data types/data structures (object lists, images, point clouds, etc.) have to be exchanged. This also includes the

[11] https://fmi-standard.org

[12] https://dcp-standard.org/

challenge of how to interpret the data. Regarding the generic test architecture, OSI provides a standardized interface for environment and environment sensor data, which can be used by automated driving functions. Hence, this interface enables the connection between function development frameworks and the simulation environment.

The OSI implementation is based on google protocol buffers[13] that provide a mechanism for serializing data which is defined in language-neutral and platform-neutral messages. OSI specifies an object-based environment description by defining messages describing the ground truth as well as the sensor data for testing in simulation environments. Ground truth data contains unmodified object data describing the environment of the ego vehicle that is the output of the simulation framework. It is based on a global reference frame. In contrast to the ground truth, sensor data describes object data in the environment relative to one specific sensor which is thus based on the sensor reference frame. This data structure contains input as well as output of statistical sensor models.

OSI is in an early development stage and thus the specification can be subject to change. Further, there are large data sets necessary to describe environment data and experiences are needed in using OSI for simulations with such large data sets.

In addition to the data structure, OSI sensor model packaging is specified that defines how the OSI sensor models have to be packaged as FMU for use in simulation environments.

During the project a simple OSI demonstrator has been set up, which shows the application of the standardized interfaces to test an Adaptive Cruise Control (ACC) function. OSI is used for the communication between the environment simulation and the function under test. The main task of the environment simulation is the generation of realistic ground truth data based on the selected scenario. The output of the environment simulation varies from general simulation data to simple and complex object lists and beyond to realistic raw sensor data. For the demonstrator "VIRES Virtual Test Drive" (VTD) is used as environment simulation software, which covers the full range from the generation of 3D content to the simulation of complex traffic scenarios. In the demonstrator mainly object lists are generated in an OSI compliant form (osi3::GroundTruth).

Model.CONNECT™ uses a TCP connection to receive the OSI ground truth data from VTD and to hand it over to a sensor model. The sensor model has been implemented as an FMU running in Model.CONNECT. For the demonstrator a simple phenomenological sensor model was implemented, which transforms the global coordinates to relative coordinates with respect to the ego car and applies a filter function to the detected objects list from the ground truth data. The filter reduces the detection range of objects based on precipitation, fog and illumination. Based on this 'real' sensor data the ACC function is tested.

To compare ground truth and the sensor data, the object lists are visualized in ROS (Robot Operating System). Therefore, OSI data is exchanged between

[13] https://developers.google.com/protocol-buffers

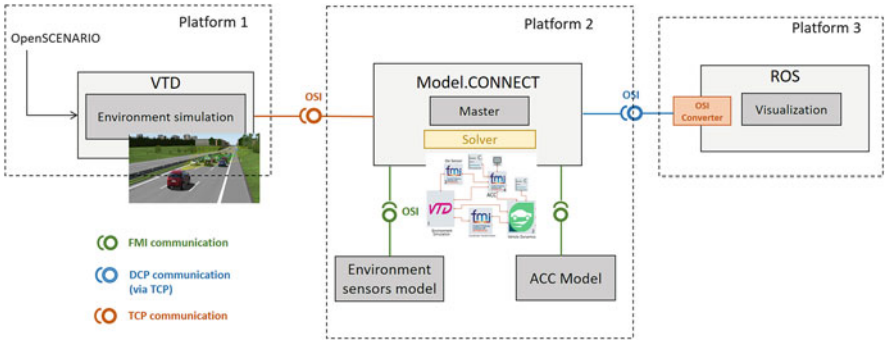

Fig. 3 Setup of the Open Simulation Interface demonstrator

Model.CONNECT and the ROS environment via DCP (Distributed Co-Simulation Protocol) over UDP. DCP is applied to evaluate this upcoming standard for distributed simulations. This standard should facilitate managing co-simulations and enables a standardized way of integrating various tools as well as real physical components. Figure 3 illustrates the demonstrator setup.

The demonstrator has mainly been used to gather experience with the specification and to assess the practical applicability. Results have been fed back into the OSI working group.

5 Simulation Model Preparation

An important ingredient for most of the test execution environments are simulation models. Simply simulating millions of test kilometers is of no value, if the simulation does not reflect the reality, at least to a certain degree. This means, the simulation is only as good as the match between the simulated signals from the sensor and vehicle model and corresponding values in the real vehicle. A model should be developed for a specific purpose (or application) and its validity determined with respect to that purpose. There are various validation techniques described in [14].

With respect to environment simulations, there are several factors, which can lead to significantly different results between tests in the real and virtual world. It starts with the modelling of the scenarios and 3D-environment, continues with material parameters of the modelled simulated objects, goes on with measuring and transferring weather conditions from real world to environment simulation and ends in the accuracy of the simulation models of sensors and ego-vehicle.

Here, we are mainly focusing on sensor models. We distinguish between generic sensor models, capable of simulating the main features of different sensor types (such as ultrasonic sensors, cameras sensors, radar sensors or LiDAR sensors), and specific sensor models used to replicate the behavior of a specific version of a sensor

from a specific manufacturer. Specific sensor models have to model the normal behavior of a sensor type as well as the peculiarities and imperfections of a specific sensor. As these imperfections often lead to a non-perfect perception of the real environment, it can lead to safety-critical situations which are the most interesting cases in vehicle system validation.

The required level of realism depends on the development phase. For early function verification, ideal sensor models are sufficient as the developed functions will have to work with sensors from different manufactures. In many cases, it is even not decided, which specific sensor instances or makes will be used in the final automated cyber-physical system.

Therefore, we need two sensor model preparation activities: first, tune the parameters of an ideal sensor model to create a specific sensor model for a specific instance of a sensor; second, validate that the specific sensor model replicates the behavior of a specific sensor in all relevant scenarios and weather conditions.

More information about sensor models and sensor model architectures is given in the respective chapters (“Radar Signal Processing Chain for Sensor Model Development”, “Camera Sensor System Decomposition for Implementation and Comparison of Physical Sensor Models”, “Functional Decomposition of Lidar Sensor Systems for Model Development”).

Here, we will spend a few more words on the challenges of sensor model parametrization. Sensor model parametrization describes the procedure to tune a generic sensor model in a way that it reflects the properties of a specific real sensor. Therefore, a lot of measurements have to be taken. First, the behavior of the sensor has to be measured. This can be done best on a proving ground, because the second important part is the measurement of the ground truth. This means that all involved traffic participants need to be instrumented with very accurate measurement devices. Only then we can determine the detailed position of each traffic participant at any time. Furthermore, all the participants of the scenario are under our control. This means that we can exactly determine their dimensions, materials, and behavior. All this information is required to reconstruct the ground truth information with the required accuracy. Another aspect which needs to be considered are environment conditions (such as weather conditions). This data is needed to calibrate the environment simulation with weather data to ensure, that different environment simulations deliver the same information to sensor models at same weather conditions. Unfortunately, no standardized metrics for environment conditions exists and it is not completely clear which metrics are required.

The basic idea of the parametrization procedure is to transfer the ground truth information into the environment simulation. The sensor models are then fed with ground truth and weather data from the simulation. At the same time the sensor model parameters are adjusted until the simulated output of the sensor model matches the measurement taken with the real sensor.

There are still a lot of unknowns for this parametrization procedure and for the final validation that the model is reflecting the reality with sufficient accuracy for the specific validation task. Nevertheless, this is a major prerequisite to make virtual

validation a useful alternative to real-world testing of automated cyber-physical systems and thus to reduce the required test effort.

6 Conclusion

In this paper we propose a seamless validation toolchain, which aims to overcome the challenges of testing automated cyber-physical systems in general and automated driving functions in concrete. The validation toolchain promotes an open and modular architecture to use different simulation models as well as to support reuse of test cases throughout various test execution environments. We highlighted that there are already first standardization activities for interfaces, which support the modular structure. Nevertheless, there are still some challenges to ensure that the simulation reflects the reality.

References

1. Koopman, P.: The Heavy Tail Safety Ceiling; Automated and Connected Vehicle Systems Testing Symposium 2018
2. Winner, H., Wachenfeld, W.: Effects of autonomous driving on the vehicle concept. In: Autonomous Driving: Technical, Legal and Social Aspects, pp. 255–275 (2016)
3. Ahmetcan, E., Kaplan, E., Leitner, A., Nager, M.: Parametrized end-to-end scenario generation architecture for autonomous vehicles. In: CEIT Conference 2018, Istanbul (2018)
4. Schneider, H., Weck, T.: Efficient active-safety-testing using advanced on-road data analysis and simulation. In: SIMVEC Conference (2018)
5. Beglerovic, H., Ravi, A., Wikström, N., Koegeler, H.M., Leitner, A., Holzinger, J.: Model-based safety validation of the automated driving function highway pilot. In: Pfeffer, P. (ed.) Proceedings of 8th International Munich Chassis Symposium 2017. Springer, Wiesbaden (2017)
6. Hanke, T., Hirsenkorn, N., Van-Driesten, C., Gracia-Ramos, P., Schiementz, M., Schneider, S.: Open simulation interface: a generic interface for the environment perception of automated driving functions in virtual scenarios: research report. http://www.hot.ei.tum.de/forschung/automotive-veroeffentlichungen/ (2017). Accessed 28 Aug 2017
7. Moten, S., Celiberti, F., Grottoli, M., van der Heide, A., Lemmens, Y.: X-in-the-loop advanced driving simulation platform for the design, development, testing and validation of ADAS. In: Intelligent Vehicle Conference (2018)
8. Schyr, C., Brissard, A.: Driving Cube—a novel concept for validation of powertrain and steering systems with automated driving. In: Advanced Vehicle Control: Proceedings of the 13th International Symposium on Advanced Vehicle Control (AVEC'16), Munich, Germany, 13–16 September 2016, p. 79. CRC Press (2016)
9. Förster, M., Hettel, R., Schyr, C., Pfeffer, P.E.: Lateral dynamics on the vehicle test bed – a steering force module as a validation tool for autonomous driving functions. In: Pfeffer, P. (ed.) Proceedings of 9th International Munich Chassis Symposium 2018. Springer, Wiesbaden (2019)
10. Gruber, A., et al.: Highly scalable radar target simulator for autonomous driving test beds. In: 2017 European Radar Conference (EURAD), Nuremberg, pp. 147–150. https://doi.org/10.23919/EURAD.2017.8249168

11. Gietelink, O.J., Ploeg, J., De Schutter, B., Verhaegen, M.: VEHIL: test facility for fault management testing of advanced driver assistance systems. In: Proceedings of the 10th World Congress on Intelligent Transport Systems and Services(ITS), Madrid, Spain, 16–20 November 2003. Paper 2639
12. T. European New Car Assessment Programme. Test protocol – AEB systems, EuroNCAP, Test Protocol 1.0, July 2013
13. Knauss, A., Berger, C., Eriksson, H., Lundin, N., Schroeder, J.: Proving ground support for automation of testing of active safety systems and automated vehicles. In: Fourth International Symposium on Future Active Safety Technology Toward Zero Traffic Accidents (FASTzero) (2017)
14. Robert, G.: Sargent, verification and validation of simulation models. In: Jain, S., Creasey, R.R., Himmelspach, J., White, K.P., Fu, M. (eds.) Proceedings of the 2011 Winter Simulation Conference. IEEE Press, Piscataway, NJ (2011)
15. Leitner, A.: Generic test architecture ENABLE-S3. Technical Report. https://www.enable-s3.eu/ (2018)