INTERNATIONAL SYMPOSIUM ON DEVELOPMENT METHODOLOGY

A Model-Based approach for automation and traceability of validation activities – clarified for Advanced Driver Assistance Systems

M.Sc. Constantin Mandel, M.Sc. Moritz Wäschle, M.Sc. Sebastian Lutz,
Dr.-Ing. Matthias Behrendt – IPEK Institute of Product Engineering at Karlsruhe Institute of Technology (KIT)

Abstract

Validation is the central activity in product engineering and a means to gain knowledge about a System-in-Development. Especially for ADAS – Advanced Driver Assistance Systems, validation is of particular importance to ensure driving safety and driving comfort. Complexity and connectivity of modern mechatronic systems like ADAS demand for new, model-based approaches for development and validation.

This contribution presents a model-based framework, consisting of methods and model interfaces, to enable automation and traceability of validation activities in product engineering. Therein, validation is regarded in the understanding of the IPEK-X-in-the-Loop-approach. The framework allows to connect models from different sources to support the formulation and assessment of validation goals as well as automating validation activities. Like this, a closed-loop to support the planning of validation activities, performing of tests and feedback and traceability of test results to the development of the product is established. The presented framework describes methods and interfaces for models to be used and can thus be used for a multitude of models and validation environments from different sources. In this contribution, the framework is exemplarily applied for validation activities for ADAS, more precisely for AEB – Autonomous/ Advanced Emergency Braking systems.

Keywords — Automated Driving, Validation, Model-Based Systems Engineering (MBSE), X-in-the-Loop (XiL)

1 INTRODUCTION AND MOTIVATION

Validation is the central activity in product engineering and the central means to gain knowledge about a SiD – System-in-Development as well as to detect errors and flaws early in the product development process [1, 2]. In this understanding, validation is not only an activity of analysis of a current state of a product. In addition, validation serves to concretize and expand goals and requirements for a product. As a consequence, validation must not be regarded as a closed phase at the end of a product development process but rather needs to start in early phases of a product development process and needs to be performed continuously into later phases of the product lifecycle [3].

Especially for ADAS – Advanced Driver Assistance Systems, validation is of significant importance as ADAS actively intervene in driving functions and thus have an influence on driving safety and comfort. On the one hand, ADAS may support in avoiding accidents like rear-end-collisions. On the other hand, malfunction of ADAS may lead to additional accidents, especially if drivers rely on their functionality. In addition, even if safe functionality of ADAS is ensured, their intervening in driving functions impacts driving comfort which might be a factor for buying decisions of potential customers.

The development and validation of ADAS, similarly to other modern mechatronic systems, is subject to rising system complexity and connectivity, internally and in relation to further systems. In order to cope with this complexity, new development approaches are necessary. Model-based approaches like MBE – Model-Based Engineering and MBSE – Model-Based Systems Engineering are widely regarded as promising approaches to cope with complexity in product engineering. Model-based approaches aim at replacing the use of a multitude of unstructured and unconnected text documents for information management with central, interconnected models.

This contribution aims at supporting a continuous validation concept in a model-based approach – clarified for ADAS. The result presented in this work is a model-based framework, consisting of modeling methods and description of model interfaces. The framework supports at establishing a traceability between models from different sources like requirements- and system models in the understanding of MBSE or simulation models as part of validation environments. With the framework, the goal-oriented formulation of validation objectives is supported. The model-based approach helps to gather interconnected information to act as a consistent background and decision support for the selection or development, respectively, of appropriate test cases and validation environments for the formulated validation objective. In addition, the framework enables to integrate virtual or mixed virtual-physical validation environments in the understanding of the IPEK-X-in-the-Loop-approach (cf. [4]). In this way, an automation of validation activities is supported. In addition, with the help of the developed framework, results from validation activities can be fed back into requirement- and system models in the understanding of MBSE to support assessment of their impact and concretize product goals and requirements. Hence, seamless validation in different phases of a product development process and for different maturity levels of a product and different validation environments can be enabled.

2 STATE OF RESEARCH AND RELATED WORK

2.1 Validation in Product Engineering and the IPEK-X-in-the-Loop-approach

Validation, as the central activity in product engineering, has a decisive role for a successful product in the market (cf. [1, 2, 4]). Validation is defined by the VDI standard 2206 as "[…] testing whether the product is suitable for its intended purpose or achieves the desired value" [5, p. 39]. In addition to validation, verification is defined as "[…] checking, whether the way in which something is realized […] coincides with the specification […]" [5, p. 38]. Thus, validation can be regarded as including verification and explicitly adding the consideration and evaluation of stakeholder expectations and -needs (cf. [4]). In this understanding, validation, in contrast to verification, is often not formally feasible and thus needs to be methodically supported (cf. [4]).

Only by validation, knowledge evolves in the product development process [1, 2]. Thus, validation enables to detect deviations from stakeholder expectations but also errors and flaws in the current state of the product. As a consequence, validation must not be regarded as a separated phase in the

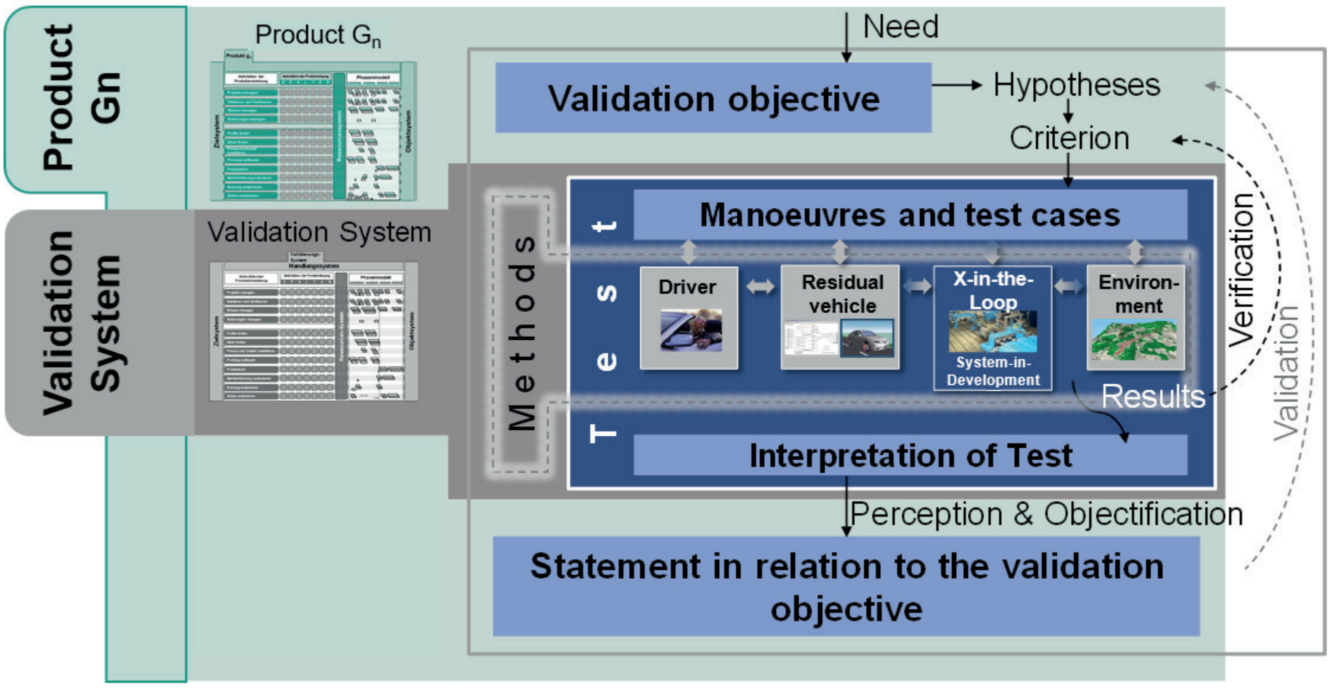


Figure 1: IPEK-XiL-approach and its relation to the product development process (adapted from [8])

product development process [3, 4]. A continuous validation concept is required, starting in early phases of the product development processes and continuing over the whole product lifecycle (cf. [3, 4]). Especially in early phases, validation offers a significant lever for product costs as costs and effort to eliminate errors significantly rise the later they are detected and processed in the product development process (cf. [6]).

In order to gain meaningful insights about a SiD, validation activities always need to be performed in a suitable validation environment taking into account the residual system/ parent systems of the SiD as well as system users and the system environment [4]. Approaches to integrate a (sub-) system in an overall system are already being described by approaches such as Model-in-the-Loop (MiL), Software-in-the-Loop (SiL) or Hardware-in-the-Loop (HiL) (cf. [4, 7]). The IPEK-X-in-the-Loop (IPEK-XiL) approach integrates those approaches and expands them regarding the requirements of a mechatronic system development with involved stakeholders of multiple (engineering) disciplines [4]. In the understanding of the IPEK-XiL-approach, all (models of) the SiD, the rest-vehicle system, the stakeholders of the system and the environment, can be realized in a physical, virtual, or mixed physical-virtual form [4]. The selection of the suitable models for the validation environment is driven by the test cases and the appropriate validation objectives for a specific validation activity [8]. The validation objective in turn is directly derived from (stakeholder) needs and knowledge gaps in the product development process [8]. Thus, statements regarding the results from validation activities should always be formulated with regard to the respective validation objective [8]. The described dependencies are schematically displayed in Figure 1.

2.2 (Model-Based) Systems Engineering

Product development nowadays faces challenges of a rising complexity and interdependency of systems. This is driven by increasing customer demands, especially regarding system functionality and reliability. At the same time, customers are expecting safe systems and have an increasing focus on environmental impacts. [9]

Traditional Systems Engineering approaches are usually document-based, meaning, that the information about a system is spread over a multitude of documents that are often just loosely or not at all connected (cf. [10]). In order to cope with this complexity and address the shortcomings of document-based development approaches, model-based approaches like Model-Based Engineering (MBE) and especially Model-Based Systems Engineering (MBSE) are being used (cf. [11]).

MBE appears to be not unambiguously defined but in general stands as an umbrella term for different model-based approaches in product development like Business Process Modeling (BPM), the use of simulation models and MBSE [12, 13].

Following the definition of INCOSE – the International Council On Systems Engineering, MBSE can be described as "[…] the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases […]" [14, p. 15]. Hence, MBSE approaches aim at replacing the multitude of unlinked documents with one or few central, traceable system models [10].

When referring to MBSE approaches, three aspects, also referred to as "the three pillars of MBSE" have to be regarded integratively [15]:

- The modeling language, defining elements and relations that can be used for modeling,
- The modeling method, describing activities and steps, how modeling is performed,
- The modeling (software) tool, i.e. the environment, in which modeling is performed.

While there is a multitude of modeling methods described in literature (see e.g. [16]) and several software tools for modeling available on the market (see e.g. [17]), SysML – the Systems Modeling Language [18] is prominently used as a language for modeling in the context of MBSE [15].

2.3 ADAS/AD Validation

The automation of driving functions is finding its way into more and more vehicles, even in low-price segments [19]. ADAS – Advanced Driver Assistance Systems do not only aim at increasing the comfort for a driver but can also ameliorate driving safety [20, 21]. Concerning driving safety, rear-end collisions are among the most frequent accidents [22]. Those collisions are often caused by driver distraction or misjudgment [22].

Two ADAS that could potentially avoid rear-end collisions are Adaptive Cruise Control (ACC) and Autonomous/Advanced Emergency Braking systems (AEB). ACC is a system for the automatic regulation of driving speed based on the traffic situation. AEB describes a system that monitors a driving situation and automatically initiates a braking maneuver in case of emergency. [23]

As ADAS actively intervene in driving situations and (at least partially) take driving duties from a driver (cf. [23]), a special focus has to be given to their validation. Therein, requirements and validation needs for ADAS may originate from legislation, consumer associations like EuroNCAP and manufacturer specific requirements to satisfy their user needs [23].

In order to be able to develop and validate ADAS cost-efficiently, the „From-Road-To-Rig" approach is being applied more and more. In this approach, the knowledge gain is shifted forward from expensive road tests to (partially) virtual environments, thus saving prototypes and increasing the number of possible tests for validation [24]. Such virtual environments are used, for example, in the context of validation activities in the understanding of the IPEK-XiL-approach (see paragraph 2.1). Due to this development, the planning and coordination of test cases and validation environments early in the product development process is becoming increasingly important. Therefore, the use of MBE approaches holds great potential.

3 RESEARCH QUESTIONS AND AIM OF WORK

As described in paragraph 2.1, validation is the central activity in product engineering. A continuous validation concept, starting in early phases and continuing over the product lifecycle, is required. The importance of validation is especially prominent in the development of ADAS, as those systems actively intervene in driving functionality and thus have a major impact on driving safety. The complexity of ADAS and their implementation in vehicles demands new, model-based validation approaches. The work presented in this paper thus investigates two research questions:

1. How can a generalized framework be created that supports continuous validation in the development of ADAS and is flexibly adaptable for different validation objectives?
2. How can different manifestations of mixed physical-virtual validation environments in the understanding of the IPEK-X-in-the-Loop-Approach be integrated in the model framework?

The focus of the work presented here is on the methodical foundations and exemplary implementations of the framework. However, the optimization of the individual models used in the framework is not in the scope of this work and is regarded in further works.

4 APPROACH

The developed model-based framework is schematically displayed in Figure 2. The framework consists of methods and model interfaces and describes a generalized approach to enable automation and traceability of validation activities. Therein, the framework bases on existing work from the authors for automating validation activities in the context of ADAS (cf. [25]). The framework can be described as consisting of three pillars (schematically separated by the dashed lines in Figure 2):

- A requirements- and system model in the understanding of MBSE,
- An automated environment for test planning and analysis of test data,
- The virtual, physical or mixed physical-virtual validation environment in the understanding of the IPEK-XiL-Approach.

The focus of this contribution lies on methods and the definition of interfaces between the three pillars. By a clear definition of those methods and interfaces, the framework may be used with a multitude of models created in various software-tools. However, the detailed (software-)

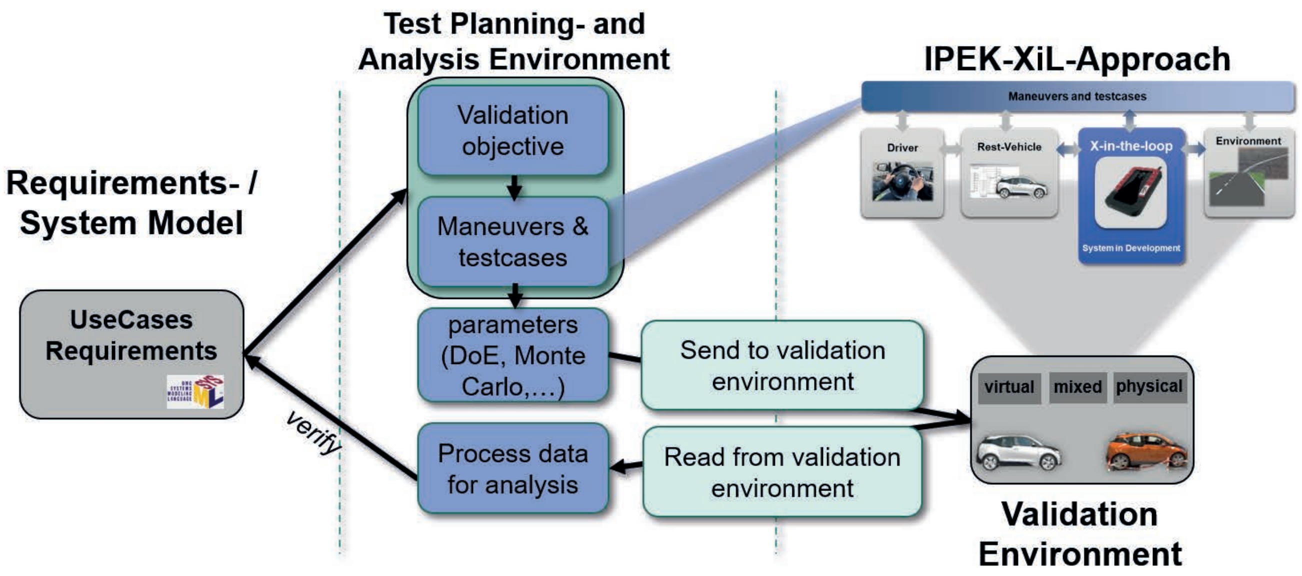


Figure 2: Schematic visualization of the model framework

implementation and optimization of the individual models is not the focus of this contribution.

The next paragraphs will discuss the individual pillars in more detail.

4.1 Requirements- and system model in the understanding of MBSE

The pillar requirements-/system model of the framework, describes SysML models in the understanding of MBSE. In the context of the framework, those models serve two purposes: Firstly, the modeled and therefore traceable information about system requirements, use cases, system design etc. serve as a consistent basis for planning test cases and test environments. Secondly, results from validation activities can be fed back into those models to assess the impact of those results and plan further steps in a product development process. A methodological approach and realization in SysML to support those two activities is described by Mandel et al. [26].

4.2 Test planning- and analysis environment

In this contribution, the authors use a federated modeling approach for the test planning- and analysis environment. In a federated (MBSE) modeling approach, models from different expert tools are linked together instead of using a single but unspecialized software tool [27, 28]. To realize the environment, the tool ModelCenter by Phoenix Integration is used as an integration platform [29]. ModelCenter allows to couple models from different sources like e.g. Matlab/Simulink [30] as well as a roller test-bench environment to a SysML-model in the understanding of MBSE. Like this, automated chains with parameter exchange of the models can be set up and analyzed.

In the test planning- and analysis environment, model parameters for maneuvers / test cases (e.g. velocity profiles for a vehicle) and the environment (e.g. specifications of the road) of the SiD are being processed based on the regarded validation objective. The parameters can either be entered manually or generated by methods such as probabilistic analysis (e.g. Monte-Carlo-simulation) or design of experiments. All parameters, are collected in a test run scenario. In order to collect the parameters and translate them in a form that is readable by the chosen validation environment (see paragraph 4.3), a Matlab script is used.

The integration of the interface to a suitable validation environment for the regarded validation goal can be realized in two ways. Firstly, the framework can be used to build up an individual test planning and execution environment for each validation environment using ModelCenter. Secondly, interfaces to different validation environments can be integrated as alternative paths in a single test planning and execution environment i.e. a single ModelCenter model. Based on the validation objective, the suited validation environment and thus the path to use in the model workflow can be manually chosen.

After performing the tests in the chosen validation environment, test output data is analyzed and post-processed. This post-processing may include simple plots or analysis of maximum or mean values for certain parameters. A special analysis case is the feedback of results into the requirements- and system (SysML) model. With the help of the SysML model, the test results can be directly compared to modeled requirements. Thus, fulfillment of requirements can be assessed for the modeled SiD.

After the analysis, the whole process can be repeated e.g. in the context of an optimization loop or a probabilistic analysis (e.g. MonteCarlo simulation). The whole process

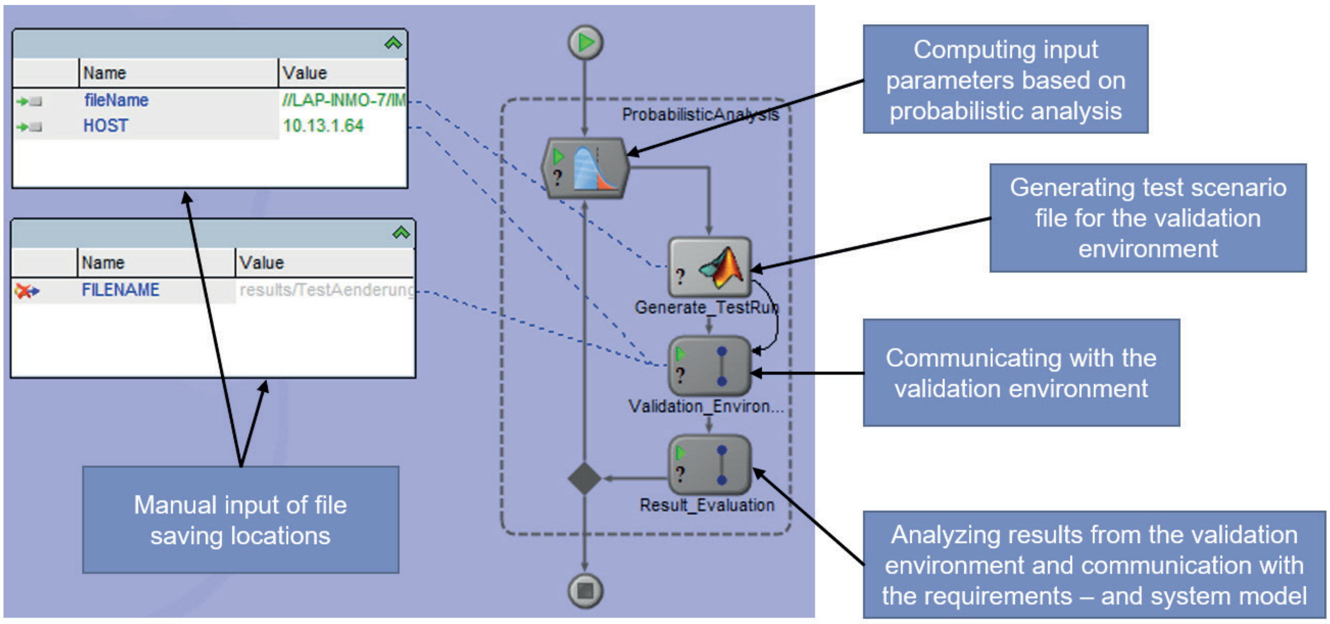

Name
Value
fileName
//LAP-INMO-7/IM
HOST
10.13.1.64
Name
Value
FILENAME
results/TestAenderung
Manual input of file saving locations
ProbabilisticAnalysis
Generate_TestRun
Validation_Environ...
Result_Evaluation
Computing input parameters based on probabilistic analysis
Generating test scenario file for the validation environment
Communicating with the validation environment
Analyzing results from the validation environment and communication with the requirements – and system model

Figure 3: Schema of the test planning and analysis

performed in the test planning and analysis environment is schematically displayed in Figure 3.

4.3 Validation environment

As described above, different validation environments can be used and integrated in the framework. Following the IPEK-XiL-approach (see paragraph 2.1), a validation environment is always oriented on the specific validation needs and validation objective to be addressed. Just like the SiD, validation environments undergo a development process and may evolve in parallel with the SiD. Validation environments considered for the model-based framework can either be in virtual, physical or mixed physical-virtual form. The validation environment interfaces directly with the test planning and analysis environment. Therefore, it has to take the processed data of parameters and scenarios as an input. This means, the preparation of this data from the test planning and analysis environment needs to be specifically programmed for each validation environment considered in the framework. Based on the input, a test run is performed in the given validation environment. The results from the test run need to be saved in a file that can again be processed by the test planning and analysis environment for further post-processing and analysis.

5 EXEMPLARY APPLICATION FOR AN ADAS USE-CASE

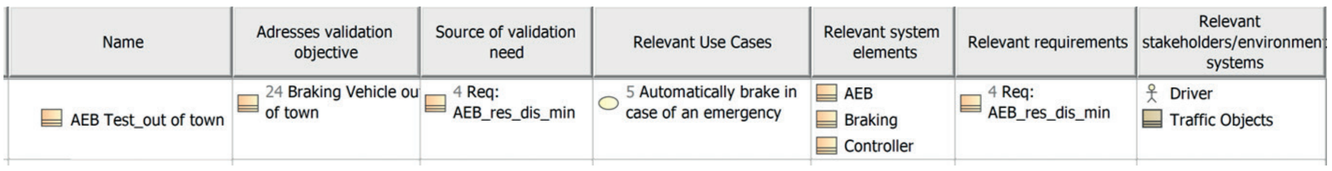
In the following section, an exemplary Case Study is presented using the developed model-based framework.

5.1 Case Study

The development of automated driving functions, as a driver assistance system or as fully automated driving, involves a large number of work steps involving very different specialist domains. For example, initial function prototypes are created in virtual test environments to test the software for functional safety. As the software matures, more and more hardware components, such as sensors and actuators, are integrated into the tests. Finally, the developers focus on integration into the overall vehicle, where the application is tested with regard to the customer experience in addition to interactions with other vehicle functions.

Throughout all these development stages, evaluation parameters show the functionality of the new function and thus document the degree of maturity.

This paper uses the development of an emergency braking assistant (AEB) system as an example to show how the model-based framework can be used to assess the fulfillment of validation needs and thus monitor the degree of maturity of a system. The following exemplary evaluation parameters can be used to observe the system behavior over different validation environments and to analyze compliance with development goals, also referred to as KPIs – Key Performance Indicators:



Name	Adresses validation objective	Source of validation need	Relevant Use Cases	Relevant system elements	Relevant requirements	Relevant stakeholders/environmen systems
AEB Test_out of town	24 Braking Vehicle ou of town	4 Req: AEB_res_dis_min	5 Automatically brake in case of an emergency	AEB Braking Controller	4 Req: AEB_res_dis_min	Driver Traffic Objects

Figure 4:Relevant information for modeled test in an AEB scenario

Detection distance

This value describes the distance at which the system recognizes an obstacle as such. From this point on, the automation has time to take steps to prevent a collision. This value is significantly influenced by the performance of the sensor system and its evaluation logic.

Deceleration distance

The deceleration distance represents the distance from which the actuators of the brakes reduce the vehicle speed. Up to this point, the AEB system has therefore detected the obstacle, assessed the situation in general and evaluated an intervention by means of deceleration with the appropriate scope as the necessary reaction.

Maximum deceleration

The maximum deceleration must be interpreted with respect to two criteria. On the one hand, vehicle occupants perceive strong, suddenly occurring delays as unpleasant, so that their build-up within the development must be set to a tolerable level by application work. On the other hand, collision with the obstacle can only be prevented by suitable deceleration. The freedom in designing the maximum deceleration is thus directly dependent on the detection distance and the deceleration distance. The earlier the obstacle is detected, the more distance is available to reduce the vehicle speed, which means that a lower maximum deceleration must be effective.

Residual distance

The residual distance results from the remaining distance to the obstacle if the vehicle speed could be completely reduced before an impact. If the automation has not been able to do this, the deceleration at contact with the obstacle is used for the functional evaluation.

5.2 Requirements- and system model

The requirements- and system model for the described case study is built up using SysML and the software Cameo System Modeler [31]. The focus of the modeled requirements is on requirements for the ACC and AEB system. As described in paragraph 2.3, those requirements may originate from externally defined regulations (e.g. mandatory standards) or from stakeholder expectations e.g. regarding driving comfort. Different modeling methods like e.g. OOSEM [32] or SYSMOD [33] are described in literature to model the requirements and establish a traceability to a modeled system architecture in SysML. Mandel et al. describe a MBSE approach, consisting of SysML language extensions, methods and a framework to explicitly support a continuous validation concept with the help of MBSE [26]. Using the approach of Mandel et al., validation objectives are modeled and linked to the already modeled system requirements and user needs. When planning a test based on a specific validation objective, the traceability established in the SysML model following the approach of Mandel et al. allows to automatically compute an overview of relevant use cases, elements of the system architecture, requirements, stakeholders and environment systems. Figure 4 exemplarily shows the computed information that is, based on the modeled information, relevant for a test "AEB Test_out_of_town". This information serves as a consistent starting point to develop appropriate test cases and choose/develop an appropriate validation environment.

5.3 Automation Process

Based on the information about a test gathered from the requirements- and system model, the test planning takes places in the test planning- and analysis environment, i.e. for the work presented in this paper ModelCenter. A pre-defined ModelCenter component for probabilistic analysis in the way of MonteCarlo simulation is chosen to compute different values for input parameters. The input parameters to be varied (e.g. initial distance from a vehicle to another vehicle), their ranges as well as the analyzed output parameters (e.g. the residual distance after emergency braking) are chosen based the information from the requirements- and system model.

The input parameters are further processed in a script that generates a driving scenario in form of a file that can be input for the chosen validation environment. The example presented in this paper bases on the works from Mandel et al. [25] to compute a simple scenario using a Matlab script. The SiD, i.e. the vehicle to be investigated, follows a target vehicle on an open, straight road out of town. The target vehicle drives at a constant speed and suddenly brakes at a certain point in time (see Figure 5). For simplicity of the example, the initial speed of the target vehicle as well as the deceleration of the target vehicle during braking are varied in the MonteCarlo simulation. The Matlab script then transforms the scenario in an input file for the simulation in the software CarMaker [34]. The file is then saved on a server.

In order to read the created test run scenario file in a validation environment, perform the tests and feedback test results into the test planning and analysis environment, a communication between ModelCenter and the validation

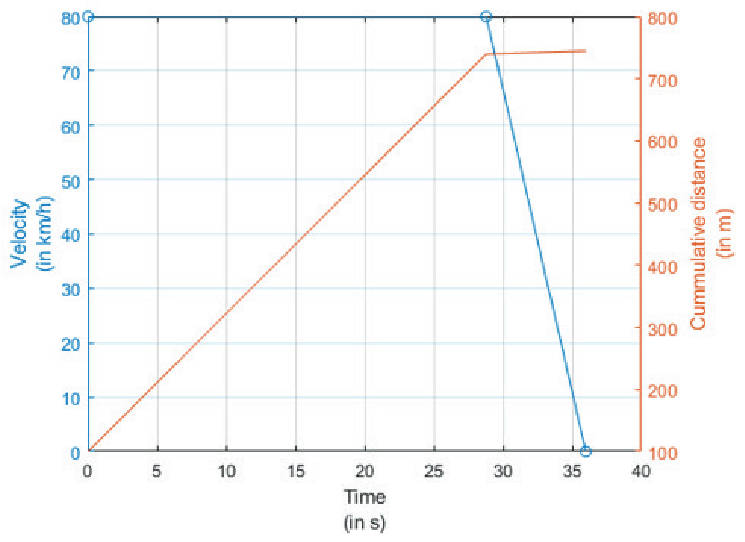


Figure 5: Exemplarily investigated driving scenario

environment is needed. As described above, the illustrating example presented in this paper uses CarMaker as the validation environment.

The connection is realized with python scripts to transmit information over the Microsoft built-in services teletype network (telnet) -client and -server, as well as network protocol. This allows to access and adjust intranet data, as well as sending information about the test, output quantities, start/stop commands for the validation and wait for a specific state.

5.4 Validation environment

The validation environment used for the example in this work is based on the software CarMaker that can either be used as a virtual simulation environment or included in a mixed physical-virtual validation environment in the understanding of the IPEK-XiL-approach. Existing works of the authors show e.g. how to use CarMaker and a roller test-bench together [24]. The roller test-bench environment consists of a simulation environment with real-time system for a rest-vehicle simulation and a BMWi3 on a roller test-bench (see Figure 6).

By loading a generated test run scenario file, the setup of the simulation environment is adapted based on the current input parameters. Further supporting scripts in the test planning- and analysis environment allow to change output parameters to be recorded as well as simulation time. The scenario can run on the real time system with the connection to the roller test-bench or only in CarMaker as a fully virtual simulation. Results of a test run in the validation environment are stored on a server.

5.5 Post-processing

Post-processing of the test results is performed in three different ways. Firstly, ModelCenter offers built-in possibilities to generate e.g. box-plots or scatter-plots. However, those plots are of course restricted to the parameters that are varied in ModelCenter.

The validation environment may produce result files, with many more parameters. Therefore, a second post-processing option, in the form of an additional Matlab script is used e.g. to draw plots of additional variables (see e.g. Figure 7). In Figure 7, an AEB scenario is shown with three parameters over time: acceleration of the investigated car center of gravity, distance to the target vehicle (e.g. determined by a camera sensor) and braking of the braking actors (1 = 100% braking).

At the beginning of the test, the investigated vehicle, hereafter referred to as ego vehicle, is standing still. After two seconds, the ego vehicle starts accelerating. The decreasing detected distance to the target after around 7 seconds leads to a decrease in acceleration and even braking. When the target vehicle performs a sudden brake after around 29 seconds and the distance to the target is getting very low, the ego car performs an AEB maneuver. This leads to braking and high deceleration. Following this maneuver up to a

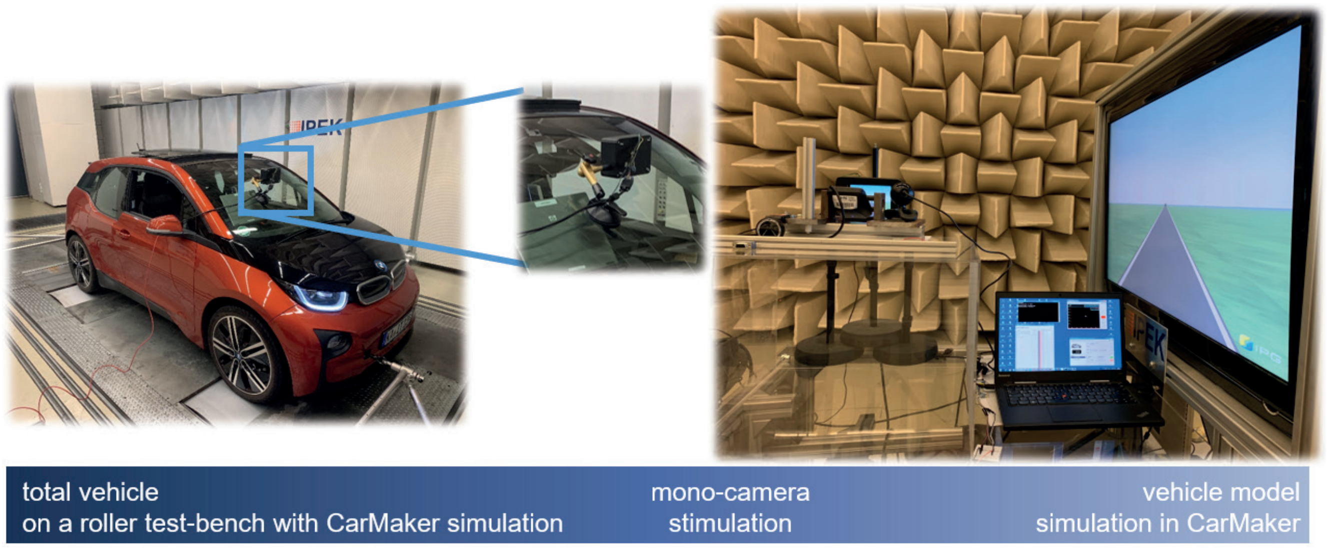


Figure 6: Validation environment for ADAS validation

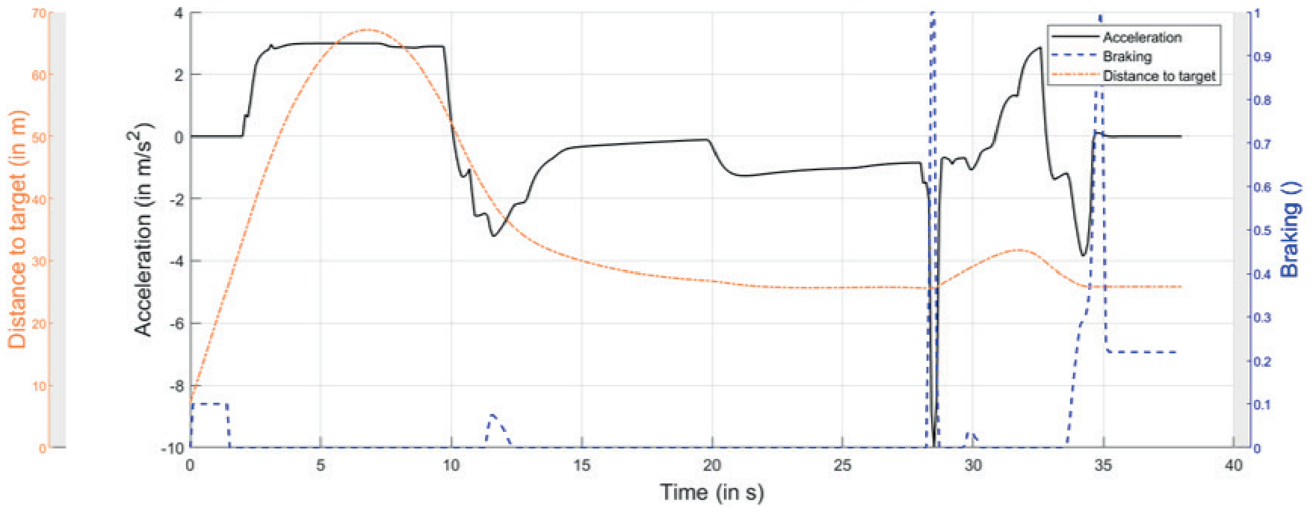


Figure 7: Plot of measured distance to a target vehicle, car acceleration and braking over time

standstill, the ego car is driving a few more meters until the scenario is terminated.

As a third way of post-processing, results from the validation environment are linked back into the SysML-model. In this way, test results can directly be compared to requirements in order to check, which requirements are satisfied with the given test parameters (see Figure 8). In addition, the traceability established in the SysML model can help to identify requirements, system elements etc. that are impacted by the test result. Figure 9 exemplarily shows a relation map for the fed back results from the AEB test. Analysis diagrams of the SysML-model like Figure 9 can then help to prioritize and plan possible targets for following development steps (e.g. changes on the chosen sensors for the AEB system).

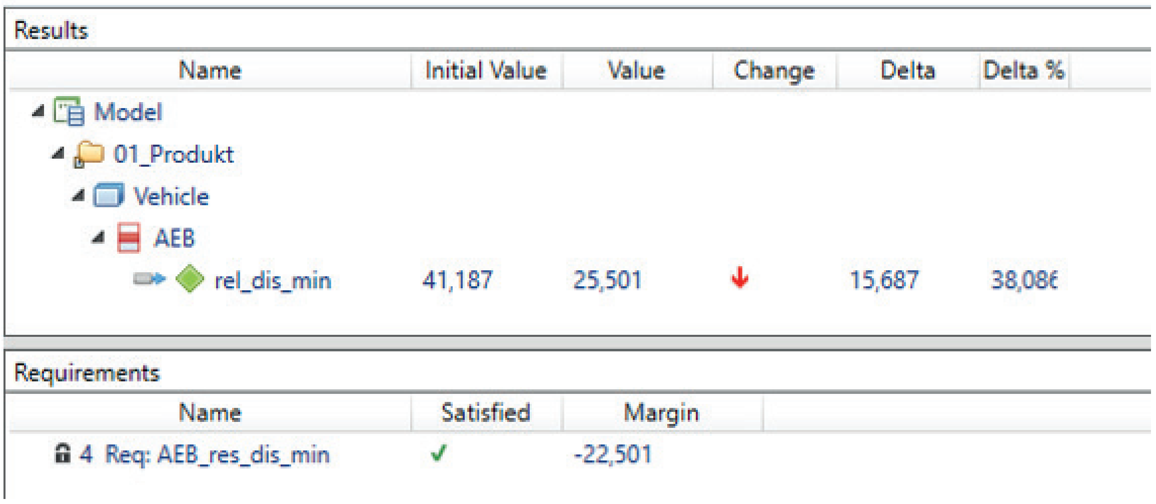
5.6 Application in Matlab environment

As described above, the developed framework can be applied for different validation environments covering different validation objectives. The framework has already been used in integration with a Matlab Simulink based simulation environment [35]. The simulation environment manages different scenarios for Adaptive Cruise Control (ACC) and Automated Emergency Braking (AEB).

For AEB the following scenarios can be realized:

1. Car-to-Car Rear stationary – City
2. Car-to-Car Rear stationary – Urban
3. Car-to-Car Rear moving – Urban
4. Car-to-Car Rear braking – Urban.

The chosen scenario, vehicle parameters, test track parameters and further road users' parameters can be adjusted



Results

Name	Initial Value	Value	Change	Delta	Delta %
Model					
01_Produkt					
Vehicle					
AEB					
rel_dis_min	41,187	25,501	↓	15,687	38,086

Requirements

Name	Satisfied	Margin
4 Req: AEB_res_dis_min	✓	-22,501

Figure 8: Analysis of requirements satisfaction based on the results from the validation environment

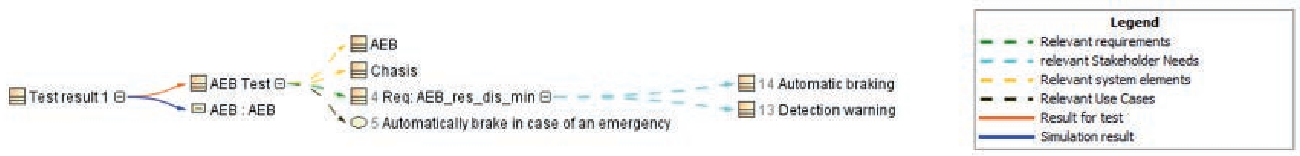


Figure 9: Using the traceability in the SysML model to assess the impact of test results

and thus made available for manipulation in ModelCenter as the test planning and analysis environment. The model of the simulation environment consists of a model for data perception and a controller model. The data perception model uses vision, radar and the curvature of the test track to calculate relative velocity and distance to the nearest target. The controller model has the input parameter relative velocity, distance and actual velocity. It calculates the set acceleration of the ego vehicle. The used model for the AEB validation environment is shown in Figure 10. [35]

6 DISCUSSION IN THE CONTEXT OF THE STATE OF RESEARCH

This contribution builds on existing work for automated validation activities of ADAS, X-in-the-Loop validation and integration inside a holistic environment (c.f. [25, 36]).

In a previous work, the authors demonstrated the feasibility of integrating a validation environment for ADAS validation with requirement- and system models [25]. In addition, the feasibility of automating validation activities in the developed environment of multiple models have been described [25].

In addition to ModelCenter, other existing software tools on the market aim at integrating different types of (virtual) models. For instance, the platform Model.CONNECT contains scenario management, RDE models and provides the interface to real ECU and HiL tests [36].

Existing MBSE methodologies like the SPES framework [37], SYSMOD [33] or FAS4M [38] with their specific focus in the fields of embedded systems, general systems design and mechanical systems design, respectively, build the background for the MBSE approach used in this paper. However, as described by Mandel et al. existing approaches do not comprehensively target the support of validation in product engineering, leading to the development of the MBSE approach used in this paper [26].

The shown approach and model-based framework might help to automatize validation activities and link different models together. Scheeren and Pereira conducted an industrial case study for the combination of Model-Based Systems Engineering, Simulation and Domain Engineering in the development of Industrial Automation Systems. The authors conclude by the use of tests, that the projects driven by models can substantially help overcome the challenges introduced by complexity. However, longer developing time and errors in linking different tools are challenges to consider [39]. The authors of this contribution agree, that the effort to build up the environments should not be neglected. Methods and processes to continuously evaluate the setup and use of the described framework based on objectives in the development and validation of ADAS need to be subject of future research. With the integration of MBSE in the product development process, especially with focus on reuse, the used models may help in multiple development processes and in further product generations.

The presented framework follows the concept of a federated modeling approach (cf. [28]) by allowing the integration of models of various types and from various sources. Furthermore, results generated by validation activities

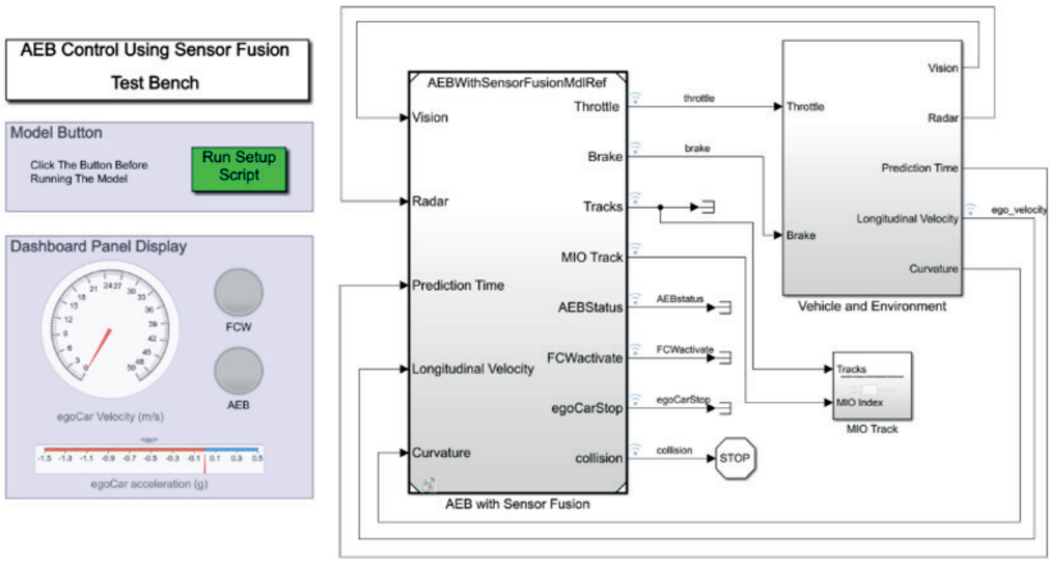


Figure 10: AEB test-bench in Matlab Simulink (cf. [35])

performed in the framework can be seamlessly and traceable integrated in a requirements- and system model. Thus, the framework offers a flexible means to support continuous and traceable validation. The focus of the research presented in this paper is on the development of the described framework and enabling the connection of various types of models from various sources. However, the internal validity of the models themselves is not part of the presented research.

7 SUMMARY AND OUTLOOK

This contribution describes a model-based framework, consisting of methods and model interfaces, to enable automation and traceability of validation activities in product development. Application of the framework to connect requirements- and system models, a test planning- and analysis environment and different validation environments with test-benches is shown. The framework allows to link test results to requirements and stakeholder needs modeled in SysML. In this way, a continuous and traceable validation in product engineering is supported. Linking realtime test-benches and simulation environments, the framework enables a need-oriented validation. The framework supports the validation process in different phases of the product development process and uses approaches such as MonteCarlo analysis or Design of Experiment and partly automated, remotely controlled expert tools.

As presented in chapter 6, extensions of the requirements- and system model (e.g. further integration of descriptions of the test-benches in the model) and of the validation environments (e.g. coverage of all scenarios required by norms and standards) should be considered in further research. Further work should also support a fully automated scenario generation, which is based on ASAM standards OpenX (e.g. OpenScenario). In addition, the integration of single hardware like a camera test-bench can be realized to achieve more functionality and a seamless variation between different validation environments.

8 REFERENCES

[1] A. Albers, “Five Hypotheses about Engineering Processes and their Consequences,” in Proceedings of the TMCE 2010, 2010.

[2] A. Albers, M. Behrendt, and S. Ott, “Validation – Central Activity to Ensure Individual Mobility,” in Automobiles and sustainable mobility: Proceedings of the FISITA 2010 World Automotive Congress, FISITA, Ed.: Gépipari Tudományos Egyesület = Scientific Association for Mechanical Engineering (GTE), 2010.

[3] A. Albers, M. Behrendt, S. Klingler, N. Reiss, and N. Bursac, “Agile product engineering through continuous validation in PGE - Product Generation Engineering,” Design Science Journal, vol. 3, 2017.

[4] A. Albers, M. Behrendt, S. Klingler, and K. Matros, “Verifikation und Validierung im Produktentstehungsprozess,” in Handbuch Produktentwicklung, U. Lindemann, Ed., München: Hanser, 2016, pp. 541–569.

[5] VDI 2206 - Entwicklungsmethodik für mechatronische Systeme: Entwicklungsmethodik für mechatronische Systeme, 2206, Verein Deutscher Ingenieure VDI.

[6] K. Ehrlenspiel and H. Meerkamm, Integrierte Produktentwicklung - Denkabläufe, Methodeneinsatz, Zusammenarbeit: Hanser, 2013.

[7] E. Bringmann and L. Krämer, Eds., Model-Based Testing of Automotive Systems: Proceedings of ICST, 2008.

[8] [8] C. Mandel, K. Wolter, K. Bause, M. Behrendt, M. M. Hanf, and A. Albers, “Model-Based Systems Engineering methods to support the reuse of knowledge within the development of validation environments,” in SysCon 2020: 14th Annual IEEE International Systems Conference, IEEE, Ed., 2020.

[9] B. Beihoff et al., “A World in Motion - Systems Engineering Vision 2025,” 2014. Accessed: Jan. 17 2019. [Online]. Available: https://www.incose.org/docs/default-source/aboutse/se-vision-2025.pdf

[10] D. D. Walden, G. J. Roedler, K. J. Forsberg, R. D. Hamelin, and T. M. Shortell, SYSTEMS ENGINEERING HANDBOOK - A GUIDE FOR SYSTEM LIFE CYCLE PROCESSES AND ACTIVITIES. Hoboken, NJ, USA: John Wiley & Sons, Inc, 2015.

[11] S. Kleiner and S. Husung, “Model Based Systems Engineering: Prinzipen, Anwendung, Beispiele, Erfahrung und Nutzen aus Praxissicht,” in Tag des Systems Engineering: Herzogenaurach, 25.-27. Oktober 2016, 2016, pp. 13–22. Accessed: Jan. 17 2019.

[12] PivotPoint Technology Corp., MBE Forum: Model-Based Engineering Visual Glossary. [Online]. Available: https://modelbasedengineering.com/glossary/ (accessed: Mar. 7 2019).

[13] Phoenix Integration, MBE: Model Based Engineering. [Online]. Available: https://www.phoenix-int.com/application/mbe-model-based-engineering/ (accessed: Sep. 16 2021).

[14] International Council on Systems Engineering (INCOSE) - Technical Operations, “INCOSE Systems Engineering Vision 2020,” 2007. Accessed: Jan. 17 2019. [Online]. Available: http://www.incose.org/media/upload/SEVision2020_20071003_v2_03.pdf

[15] L. Delligatti, SysML distilled: A brief guide to the systems modeling language: Upper Saddle River, NJ; Munich [u.a.] : Addison-Wesley, 2014.

[16] J. A. Estefan, “Survey of model-based systems engineering (MBSE) methodologies,” Incose MBSE Focus Group 25 (8), 2007. Accessed: Sep. 16 2021. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.517.219&rep=rep1&type=pdf

[17] T. Weilkiens, MBSE Tools: Popular SysML/MBSE Modeling Tools. [Online]. Available: https://mbse4u.com/sysml-tools/ (accessed: Sep. 16 2021).

[18] Object Management Group (OMG), “OMG Systems Modeling Language SysML 1.6,” Nov. 2019. Accessed: May 20 2021. [Online]. Available: https://www.omg.org/spec/SysML/About-SysML/

[19] J. Gorzelany, The Cheapest Cars And SUVs With Safety-Minded Self-Driving Technology. [Online]. Available: https://www.forbes.com/sites/jimgorzelany/2017/10/23/the-cheapest-cars-and-suvs-with-self-driving-technology/?sh=6689cf0b464a

[20] VDA-Verband der Automobilindustrie e.V., “Automatisierung: Von Fahrerassistenzsystemen zum automatisierten Fahren,” VDA-Verband der Automobilindustrie e.V., Berlin, Sep. 2015. Accessed: Aug. 10 2018. [Online]. Available: https://www.vda.de/dam/vda/publications/2015/automatisierung.pdf

[21] ADAC e.V., “Information zu neuen Fahrzeugsystemen zur Erhöhung der Verkehrssicherheit (General Safety Regulation 2019): Verordnung über die Typgenehmigung von Kraftfahrzeugen im Hinblick auf ihre allgemeine Sicherheit und den Schutz der Fahrzeuginsassen und von ungeschützten Verkehrsteilnehmern,” München, Jan. 2020. Accessed: Sep. 16 2021. [Online]. Available: https://assets.adac.de/image/upload/v1593170134/ADAC-eV/KOR/Text/PDF/TO_GSR_Final_2020_ofecss.pdf

[22] EuroNCAP, “European New Car Assessment Programme (Euro NCAP): Test Protocol - AEB systems,” Version 1.1, 2015. Accessed: Mar. 11 2019. [Online]. Available: https://cdn.euroncap.com/media/17719/euro-ncap-aeb-test-protocol-v11.pdf

[23] H. Winner, S. Hakuli, F. Lotz, and C. Singer, Handbuch Fahrerassistenzsysteme: Grundlagen Komponenten und Systeme für aktive Sicherheit und Komfort, 3rd ed. Wiesbaden: Springer Vieweg, 2015.

[24] S. Lutz, M. Behrendt, and A. Albers, “Continuous development environment for the validation of autonomous driving functions,” in 20. Internationales Stuttgarter Symposium: Automobil- und Motorentechnik, Stuttgart, 2020.

[25] C. Mandel, S. Lutz, O. Rau, M. Behrendt, and A. Albers, “Model-Based Engineering für die Automatisierung von Validierungsaktivitäten am Beispiel Fahrerassistenzsysteme,” in EEE-Entwerfen Entwickeln Erleben 2019-Band 1, Dresden, 2019. Accessed: May 20 2021.

[26] C. Mandel, J. Böning, M. Behrendt, and A. Albers, “A Model-Based Systems Engineering Approach to Support Continuous Validation in PGE - Product Generation Engineering,” in IEEE ISSE International Symposium on Systems Engineering 2021, 2021. Accessed: Sep. 14 2021.

[27] R. Peak, C. Paredis, L. McGinnis, S. Friedenthal, and R. Burkhard, “Integrating System Design with Simulation and Analysis Using SysML,” INSIGHT, vol. 12, no. 4, pp. 40–44, 2009, doi: 10.1002/inst.200912440.

[28] S. Kleiner, Föderatives Informationsmodell zur Systemintegration für die Entwicklung mechatronischer Produkte. Zugl.: Darmstadt, Techn. Univ., Diss., 2003. Aachen: Shaker, 2003.

[29] Phoenix Integration, Phoenix Integration ModelCenter. [Online]. Available: https://www.phoenix-int.com/ (accessed: Mar. 7 2019).

[30] MathWorks, MATLAB. [Online]. Available: https://de.mathworks.com/products/matlab.html (accessed: Mar. 11 2019).

[31] No Magic Inc., Cameo Systems Modeler. [Online]. Available: https://www.nomagic.com/products/cameo-systems-modeler (accessed: May 27 2019).

[32] S. Friedenthal, A. Moore, R. Steiner, and Lockheed Martin, The MathWorks, Raytheon, A practical Guide to SysML: Morgan Kaufmann, 2012.

[33] T. Weilkiens, SystemsEngineering mit SysML/UML Modellierung, Analyse, Design. Heidelberg: dpunkt.verlag GmbH, 2006.

[34] IPG Automotive GmbH, CarMaker. [Online]. Available: https://ipg-automotive.com/de/produkte-services/simulation-software/carmaker/ (accessed: Mar. 11 2019).

[35] B. Zeng, “Entwicklung und Implementierung eines durchgängigen Model-Based Engineering Ansatzes für die Validierung technischer Systeme am Beispiel Fahrerassistenzsysteme,” Unveröffentlichte Bachelor-/Masterarbeit, IPEK - Institut für Produktentwicklung, Karlsruher Institut für Technologie (KIT), Karlsruhe, 2020.

[36] AVL List GmbH, Model.CONNECT. [Online]. Available: https://www.avl.com/documents/10138/2699442/Model.CONNECT%E2%84%A2+Overview (accessed: Feb. 2 2021).

[37] K. Pohl, M. Broy, H. Daembkes, and H. Hönninger, Advanced Model-Based Engineering of Embedded Systems: Extensions of the SPES 2020 Methodology. Cham: Springer International Publishing, 2016. [Online]. Available: http://gbv.eblib.com/patron/FullRecord.aspx?p=4755382

[38] G. Moeser, M. Grundel, T. Weilkiens, S. Kümpel, C. Kramer, and A. Albers, "Modellbasierter mechanischer Konzeptentwurf: Ergebnisse des FAS4M-Projektes," in Tag des Systems Engineering, 2016.

[39] I. Scheeren and C. E. Pereira, "Combining Model-Based Systems Engineering, Simulation and Domain Engineering in the Development of Industrial Automation Systems: Industrial Case Study," in 2014 IEEE 17th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing.