Methods for the Development of Collaborative Embedded Systems in Automated Vehicles

© FEV Europe GmbH

In future automated vehicles, collaborative embedded systems will work in system groups within a dynamic context that poses a particular challenge for design and test concepts. The complex development process must be supported by a heterogeneous tool environment. In the CrESt research project, RWTH Aachen University and FEV Europe are pursuing the goal of establishing interoperability between different tools. In this paper, four development types and tool methods are presented which address these challenges.

INTRODUCTION

Driving assistance systems already have a long history. Since the 1950s, for example, cruise control to keep a vehicle traveling at a constant speed has played an increasingly important role. With Adaptive Cruise Control (ACC), today's vehicles can follow others at a predefined distance by controlling the longitudinal dynamics with the aid of sensors. Collaborative Adaptive Cruise Control (CACC) is an advancement of this technology, enabling prospective vehicles to adapt their speed to others in front with direct Vehicle-to-Vehicle (V2V) communication. CACC systems can react more directly to speed fluctuations of the vehicle in front,

[MGWJ20] M. Meyer, C. Granrath, L. Wachtmeister, N. Jäckel:
Methods for the Development of Collaborative Embedded Systems in Automated Vehicles.
In: ATZelectronics worldwide, 15(12), pp. 58–63, Springer, Dec. 2020.
www.se-rwth.de/publications/

AUTHORS

Max-Arno Meyer, M. Sc.
is Research Assistant within the Junior Professorship for Mechatronic Systems for Combustion Engines at the RWTH Aachen University (Germany).

Christian Granrath, M. Sc.
is Research Assistant within the Junior Professorship for Mechatronic Systems for Combustion Engines at the RWTH Aachen University (Germany).

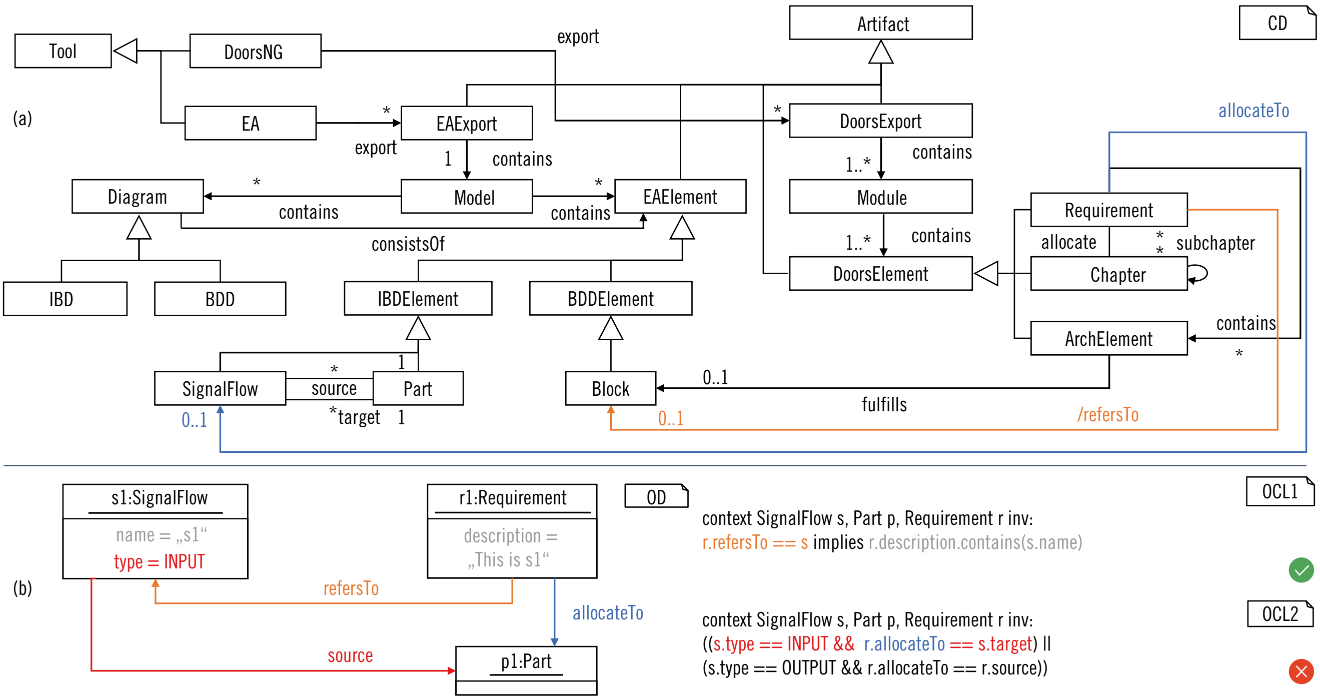
Louis Wachtmeister, M. Sc.
is Research Assistant Software Engineering at the RWTH Aachen University (Germany).

Dr. Nicolas Jäckel
is Systems Engineer for Intelligent Mobility and Software at FEV Europe GmbH in Aachen (Germany).

even if it is outside the line of sight, due to the absence of technically induced dead times. These new types of systems do not exclusively use sensor data for operation, but are increasingly equipped with open interfaces and include communication and collaboration beyond the vehicle (Vehicle-to-everything, V2X). They form so-called Collaborative Embedded Systems (CES). When these systems collaborate, they can communicate to achieve strategic goals at a higher level in a so-called Collaborative System Group (CSG). The development of CES is subject to characteristic challenges, such as operating in an open context that changes dynamically in an unpredictable manner at design time. The systems must work together and be able to change their behavior at runtime. High security risks exceed the level of conventional systems whose design and test concepts here are insufficient. Therefore, new methods for the development of collaborative embedded systems have to be found. In this paper four development and tool methods are presented that address these challenges.

REFERENCE ARCHITECTURES FROM CES

If the context is static, a system architecture is defined and developed in advance. In a dynamically changing CSG with additional environmental influences, reference architectures of the respective system of the group can be used. Matches between different designs with the current reference architecture can be identified automatically. Components of static system architectures without an extrinsic match are assigned automatically to a new reference architecture. Due to the higher level of complexity by designing new functions, this approach cannot be transferred directly to the development of a dynamic open system. The step-by-step development based on a specific context with re-use of a common reference architecture con-



```
context SignalFlow s, Part p, Requirement r inv:
r.refersTo == s implies r.description.contains(s.name)
```

```
context SignalFlow s, Part p, Requirement r inv:
((s.type == INPUT && r.allocateTo == s.target) ||
(s.type == OUTPUT && r.allocateTo == r.source))
```

FIGURE 1 Artifact model (a) and extracted object diagram with OCL conditions for analysis (b) (© RWTH Aachen University) taining only draft objects represents a possible method. Systems are first developed for specific static contexts; then open systems are extracted. A draft of the reference architecture is thus defined in advance and should then be used for any dynamic development [1].

ARTIFACT MODELING

The development of CES increases the number of interdependent developmental artifacts. An artifact is any individually storable unit that can be identified by a unique name and serves a specific purpose in the project context. Respective documents specify relevant requirements that a system to be developed must fulfill, while models based on Systems Modeling Language (SysML) describe the logical and technical architecture of a system and the behavior of its components.

Artifact-based modeling offers the possibility to analyze the artifacts and their relationships that arise in the course of a project. For this purpose, a project-specific Artifact Model (AM) is created, which defines all relevant Artifact Data (AD) as well as the associated relationship types of the project. The AD structured in this way contains information about the relevant artifacts and their relations at a certain point in time.

In order to demonstrate the applicability of the approach, the artifact-based analysis process presented in [2] was refined and applied in a case study. For this purpose, the cross-tool development of requirements and system architectures in model-based projects was examined using the development tools Doors Next Generation (Doors NG) and Enterprise Architect (EA) as examples and their interrelationships were modeled, **FIGURE 1** (a). Based on this AM, extractors for exports from EA according to the standard Metadata Interchange (XMI) and exports in Requirements Interchange Format (ReqIF) from Doors NG were developed to create object diagrams. These can be checked against modeling guidelines formulated in the Object Constraint Language (OCL). **FIGURE 1** (b) shows an object diagram and OCL conditions that check whether requirements for a signal flow contain the name of the object diagram (OCL1) and whether input flows address a destination and output flows address a source (OCL2).

TEST STRATEGY

A possible method for deriving a test strategy for CES in vehicles is based on the risk assessment of both system features and operating scenarios to be considered. In order to make the coverage manageable, tests are prioritized for the large number of possible scenarios. For individual features, applied test methods, test sequences and end-of-test criteria are defined separately based on risk categories. The categories are determined using the concept of acceptable residual risk in accordance with ISO 26262 and differentiated for the following different objectives:
- functional safety (risk classification based on Automotive Safety Integrity Levels (ASIL))
- Safety of the Intended Functionality (SOTIF)
- non-functional aspects like reliability
- information security.

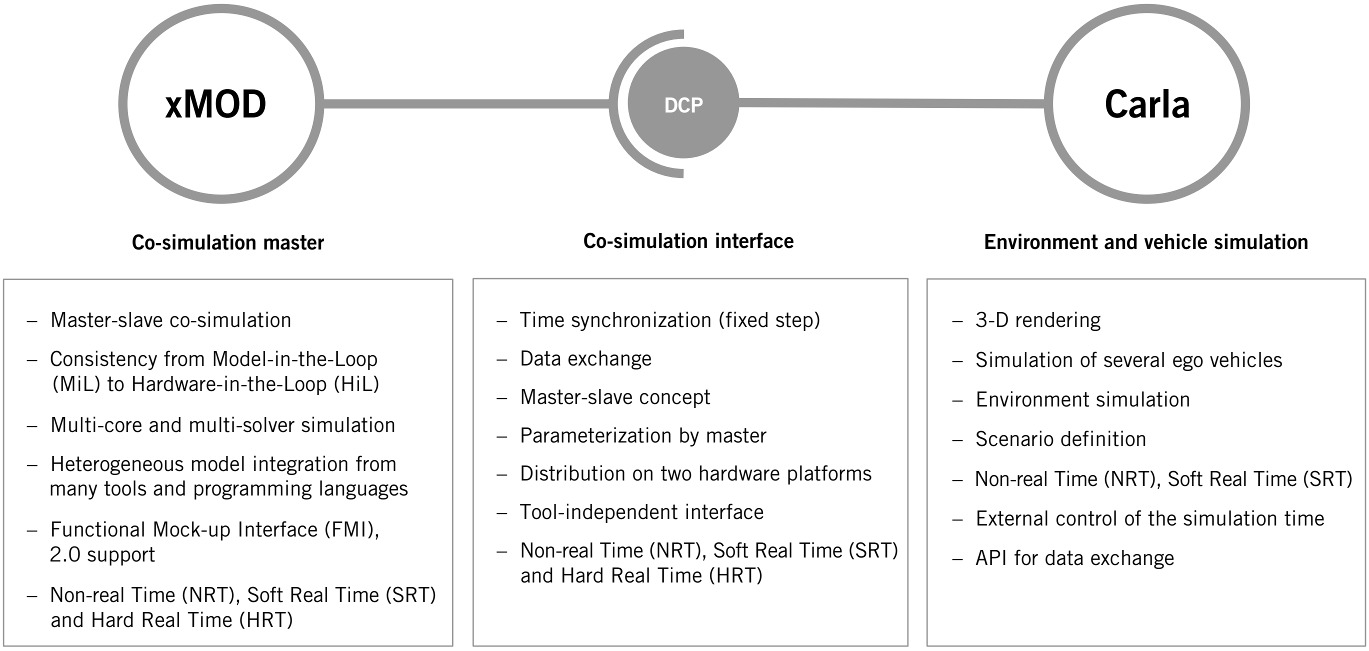
Within the scope of the risk analysis, the probability of a failure for each feature is assessed based on the complexity of the feature and context, as well as the risk to the above objectives in the event of a failure. This is carried out by means of

(a)	**Impact (maximum value)**												**Probability of errors (mean value)**							
Functional safety	Performance				Usability				Information security				Functional complexity							
ASIL rating	Real time requirements?		Time criticality?		Customer interface?		Driver interaction?		V2X communication?		Encryption necessary?		Activity and requirement complexity		Path complexity		Number of actions		Context complexity	
ASIL*	Yes	A	Yes	B	Yes	A	Yes	B	Yes	B	Yes	A	High	1	High	1	>15	1	High	1
	No	C	No	C	No	C	No	C	No	C	No	C	Medium	2	Medium	2	5-15	2	Medium	2
													Low	3	Low	3	<5	3	Low	3

*ASIL rating considered separately with regard to the test strategy

(b)	**Impact (maximum value)**											**Probability of occurence (mean value)**	
		General system load in scenario						Impact on performance					
Scenario complexity		Number of vehicles in CSG		System robustness		Context scope and diversity		Performance		…	…	Probability of scenario occurence	
High	A	>5	A	High	A	High	A	High	A	…	…	High	1
Medium	B	2-5	B	Medium	B	Medium	B	Medium	B	…	…	Medium	2
Low	C	<2	C	Low	C	Low	C	Low	C	…	…	Low	3

TABLE 1 Exemplary risk assessment charts for feature (a) and scenario (b) (© FEV Europe GmbH)



Co-simulation master

- Master-slave co-simulation
- Consistency from Model-in-the-Loop (MiL) to Hardware-in-the-Loop (HiL)
- Multi-core and multi-solver simulation
- Heterogeneous model integration from many tools and programming languages
- Functional Mock-up Interface (FMI), 2.0 support
- Non-real Time (NRT), Soft Real Time (SRT) and Hard Real Time (HRT)

Co-simulation interface

- Time synchronization (fixed step)
- Data exchange
- Master-slave concept
- Parameterization by master
- Distribution on two hardware platforms
- Tool-independent interface
- Non-real Time (NRT), Soft Real Time (SRT) and Hard Real Time (HRT)

Environment and vehicle simulation

- 3-D rendering
- Simulation of several ego vehicles
- Environment simulation
- Scenario definition
- Non-real Time (NRT), Soft Real Time (SRT)
- External control of the simulation time
- API for data exchange

FIGURE 2 Features of the interoperability interface between xMOD and Carla (© RWTH Aachen)

defined metrics, which are determined on the basis of model-based and textual requirements, implemented software and expert surveys, **TABLE 1** (a) [3]. For the risk assessment of scenarios, the probability of their occurrence and their probable impact on the CES or CSG are evaluated on the basis of complexity and relevance in relation to the defined objectives, **TABLE 1** (b).

CO-SIMULATION FOR VIRTUAL TESTS

Virtual tests of collaborative driving functions require highly complex system models. One or more vehicles with their perceptual systems are mapped in their respective contexts to analyze the collaboration behavior with respect to the strategic goals of each individual CES and CSG [4]. For the realization of such models, different cyber-physical simulation domains such as kinematics, electromagnetism, network simulation and 3-D rendering, which are mapped by isolated, specialized tools, are combined by co-simulation. Since some models, especially 3-D environment models, are bound to their executing engine, these cannot be exported (for instance, according to the Functional Mock-up Interface (FMI) standard) and executed in a co-simulation master.

Nevertheless, such models can be integrated into a co-simulation via an interoperability interface, which is implemented according to the Distributed Co-simulation Protocol (DCP) standard, **FIGURE 2**. DCP is developed as an open accompanying standard to FMI and regulates the distribution of models to different platforms independent of the communication medium [5]. Here the control of the environment and driving simulator Carla, which is based on open source, is realized by xMOD of FEV Software and Testing Solutions, whereby in this co-simulation xMOD is the DCP master and Carla the DCP slave. Configuration data for the control system are embedded in an XML file and provided to xMOD for setting up the test scenario. Carla is controlled according to the DCP state machine and data model via Protocol Data Units (PDU).

DCP allows the control of the progress in simulation time and the definition of the operating modes Hard Real Time (HRT), Soft Real Time (SRT) or Non-real-time (NRT), if supported by the slave and the communication medium [5]. The standardization of the interoperability interface according to DCP allows, in the specific application case, the outsourcing of Carla to a dedicated workstation to accelerate Model-in-the-Loop (MiL) simulation and guarantees an easier exchangeability of the tools. Via the implemented interface, several ego vehicles can be simulated in Carla in a common environment and controlled via CES controllers integrated in xMOD, **FIGURE 3** (left).

In addition, models for mapping V2V communication and sensor technology are integrated into the co-simulation utilizing FMI, allowing various Systems Under Test (SUT) and test scenarios to be realized. Among other things, the evaluation of CES and CSG for CACC on highways as well as in urban environment was thus carried out, **FIGURE 3** (right).

CONCLUSION

Methods for developing traditional embedded systems cannot be applied to CES. Therefore, in the setting of this article four approaches have been presented that take into account the dynamic context that applies to these systems. The first one is based on the step-by-step design of a reference architecture to consider relevant contexts. The second, artifact-based modeling, allows to analyze the resulting artifacts and their relationships and to evaluate them on a metrics-based foundation. The third possible test strategy for CES in vehicles is based on a risk assessment of the

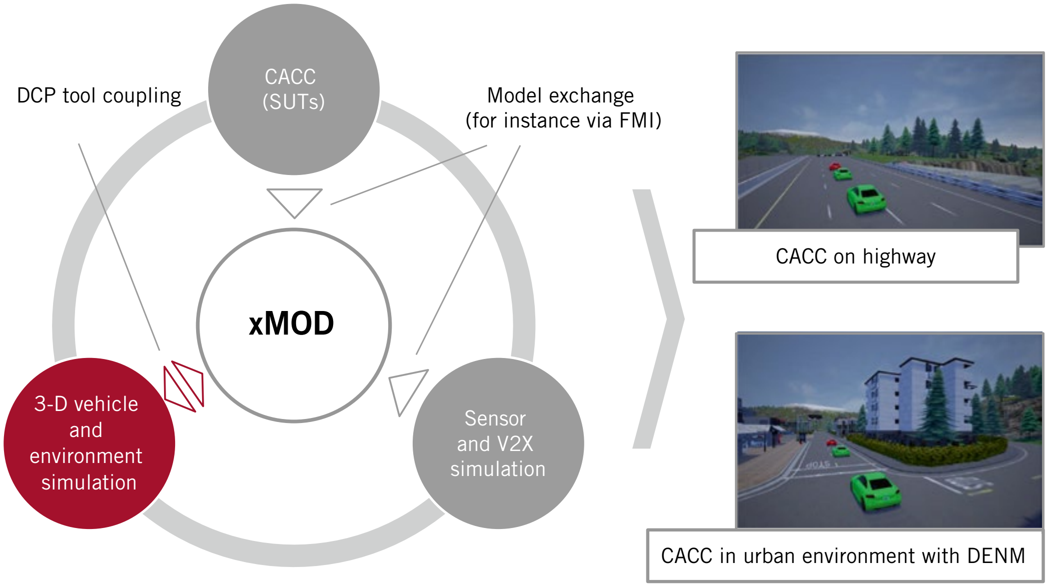


FIGURE 3 Schematic of the use of DCP tool coupling in a co-simulation platform for CACC (left); CACC on a highway and additionally with Decentralized Environmental Notification Messages (DENM) in urban environment (right) (© RWTH Aachen)

system features and allows for quality assurance under consideration of a variety of possible operating scenarios. The fourth approach is a co-simulation environment for testing CESs and CSGs in virtual environments.

REFERENCES

[1] Schlie, A.; Schulze, S.; Schaefer, I.: Comparing Multiple MATLAB/Simulink Models Using Static Connectivity Matrix Analysis. In: Proceedings of the 34th IEEE International Conference on Software Maintenance and Evolution, ICSME (2018), pp. 160-171

[2] Butting, A.; Greifenberg, T.; Rumpe, B.; Wortmann, A.: On the Need for Artifact Models in Model-Driven Systems Engineering Projects. In: Software Technologies: Applications and Foundations. In: STAF (2017), Vol. 10748, pp. 146-153

[3] Kugler, C.; Kowalewski, S.; Richenhagen, J.; Maquet, R.; Schloßer, A.: Metrics-based strategies for quality assurance of automotive embedded software. In: Bargende M.; Reuss, H.-C.; Wiedemann, J. (eds.): 17. Internationales Stuttgarter Symposium Automobil- und Motorentechnik. Wiesbaden: Springer Fachmedien Wiesbaden, pp. 711-730

[4] Kaths, J.; Meyer, M.; Granrath, C.; Andert, J.; Christiaens, S.: Virtual test drives with multiple vehicles under test for the evaluation of collaborative assisted and automated driving functions. 6th International ATZ Conference | Hybrid event, Automated Driving 2020, Wiesbaden, 2020

[5] Modelica Association Standard: Distributed Co-Simulation Protocol (DCP). Specification Document 1.0.0, March 2019. Online: https://dcp-standard.org/assets/specification/DCP_Specification_v1.0.pdf, access: October 05, 2020

THANKS

This work was partly funded by the Federal Ministry of Education and Research (BMBF) within the project “CrESt,” 01IS16043.


adhesion ADHESIVES + SEALANTS
The Trade Journal for Industrial Adhesives and Sealants
+
Adhesives Technology Compendium