Noname manuscript No.
(will be inserted by the editor)

Simulation Methods and Tools for Collaborative Embedded Systems

With focus on the Automotive Smart Ecosystems

Emilia Cioroaica · Florian Pudlitz · Ilias Gerostathopoulos · Thomas Kuhn

Received: date / Accepted: date

Abstract Embedded Systems are increasingly equipped with open interfaces that enable communication and collaboration with other embedded systems. Collaborative Embedded Systems (CES) can be seen as an emerging new class of systems which, although individually designed and developed, can form collaborations at runtime. When embedded systems collaborate with each other, functions developed independently need to be integrated for performing evaluation of the resulting system in order to discover unwanted side-effects. Traditionally, early-stage validation and verification (V&V) of systems composed of collaborative subsystems is performed by function integration at design time. Simulation is used at this stage to verify system's behaviour in a predefined set of test scenarios. In this paper we provide a survey of simulation methods and tools for the V&V of CES. In the context of one use case from the automotive domain (vehicle platooning) we present solutions (methods and tools) and challenges brought by evaluating vehicle collaboration using simulation.

Keywords Simulation · Collaborative Embedded Systems · Automotive

Emilia Cioroaica
Fraunhofer Platz 1, Kaiserslautern, 67663, Germany
Tel.: +49 631 / 6800 2242
Fax: +49 631 / 6800-9 2242
E-mail: emilia.cioroaica@iese.fraunhofer.de

Florian Pudlitz
Ernst-Reuter-Platz 7, 10587 Berlin, Germany
E-mail: florian.pudlitz@tu-berlin.de

Ilias Gerostathopoulos
Boltzmannstrasse 3, 85748 Garching bei München, Germany
E-mail: ilias.gerostathopoulos@tum.de

Thomas Kuhn
Fraunhofer Platz 1, Kaiserslautern, 67663, Germany
E-mail: thomas.kuhn@iese.fraunhofer.de

1 Introduction

More and more embedded systems are starting to be equipped with open interfaces that enable communication and collaboration with other embedded systems. This effectively enables the development of an emerging new class of systems which, although potentially individually designed and developed, can form collaborations at runtime. Following the terminology of the CrESt project [1], we call this class *Collaborative Embedded Systems* (CES) and the collaborations they form *Collaborative System Groups* (CSG). CES feature a number of advanced collaborative functionalities, such as forming a platoon of vehicles in a highway or splitting the cleaning/serving area for robots in a factory. Each such functionality is very difficult to design, implement, test, and deploy. The problem lies with the open-endedness of the system and particularly with the large number of cases one needs to consider: What if there are no available collaborators or more than needed? What if a collaborator that a subsystem depends on gets out of reach due to network errors? What if a malicious collaborator appears?

In this paper, we focus on the design and testing of CES and, following the traditional approach of performing early-stage validation and verification (V&V) via simulations, we provide a survey of existing simulation approaches and investigate whether and how simulation approaches should be extended to better address the V&V challenges brought by CES. We focus on simulation *methods*, i.e. systematic procedures that involve simulations for the purposes of V&V, and *tools*, i.e. technical frameworks and environments that support the simulation methods.

1.1 Motivation

In the traffic domain a whole range of simulation tools and methods have been successfully applied, from macroscopic traffic simulation with a global overview of all road users [23] to microscopic driving simulation [30]. Moreover, the breakthrough of virtual reality with egocentric view [45] and hybrid simulation-based test methods with real driving data [11] aim to bring significant changes in validating autonomous driving systems. Existing approaches have been mainly used for V&V of single embedded systems.

Future use cases such as vehicle platooning require embedded systems placed in different vehicles to collaborate with each other. The collaboration between systems aims to satisfy the business goals of actors introducing these systems on the market. Examples of such goals in the vehicle platooning use case are "reducing fuel consumption" or "decongestion of traffic".

Another important attribute of simulation methods is that they can be used in various lifecycle phases. Traditionally, simulation has been used for early validation of design decisions, communication protocols, and implemented business logic. Since CES are more flexible and dynamic, we envision that simulation methods will be more and more used even after deployment, i.e. at runtime, in order for systems to validate the results of their runtime decisions (e.g. joining a platoon). Approaches such as digital twins already point towards continuous simulation-based validation [37]. In [16] we have introduced a novel method that envisions use of simulation for runtime evaluation.

In short, simulation approaches need to support CES with their distinct characteristics and challenges. Given the whole range of specialized simulation methods and tools used in V&V of single embedded systems, in this paper we are interested in identifying the most suitable ones for use in V&V of CES.

1.2 Contribution

The contribution of this paper lies in providing a survey of simulation methods and tools for simulation-based V&V of CES together with a detailed discussion on the findings and future research directions. As a special theme of our survey, we specifically focus on the automotive domain due to the growing importance and relevance of collaborative vehicle use cases. We thus first describe the existing simulation methods and tools and how they already address specific challenges of CES in general and in the automotive domain in particular, and then elaborate on the remaining challenges that can lead to future research and development activities.

The paper is organized as follows: Section 2 gives an overview of embedded systems, CESs, and automotive smart ecosystems and presents the vehicle platooning use case. Section 3 explains the research method we followed in our survey. Section 4 presents different methods and tools used for the simulation of CES, while Section 5 describes simulation methods and tools used in the automotive domain. Section 6 presents an overview of challenges to be addressed in the simulation-based V&V of automotive CES, and Section 7 concludes with a summary.

2 Background

2.1 Simulation for Verification and Validation

Software functions are tested at different test levels. The software goes through unit tests, integration tests, system tests and operational acceptance tests. Due to the growing complexity of the software, the requirements for the individual test levels are also increasing. Through the use of simulations, the behaviour of complex systems can be mapped virtually and facilitates a fast, inexpensive and efficient analysis of the system. The use of simulations supports all test levels.

Usage of simulation methods for behavioural evaluation of software systems and system components has multiple benefits, independent of the domain:

- Given a set of concrete scenarios of complex interactions, simulation methods are more exploratory than analytical methods. Precision in exploration is achieved through coupling of multiple, detailed simulation models, and flexibility of exploration is achieved by simulating a wide range of behaviours, including faulty ones.
- Statistical considerations show that an estimated 10^8 to 10^9 test kilometers have to be covered to ensure the functional safety of driver assistance systems [43]. Since simulation models are digital entities, the process of running multiple simulation scenarios can be automated and parallelized.
- By coupling components or systems, systems of systems that did not exist before are formed. As a consequence, there might not be sufficient knowledge and experience about such systems. Effects of function interaction for such system can be discovered through simulation of components behaviour.
- In the automotive domain, standards like ISO 26262 [3] explicitly recommend simulation as a quality assurance technique.

2.2 Automotive Smart Ecosystems

Digital ecosystems resemble biological ecosystems. We can distinguish between two types of digital ecosystems: Software Ecosystems and Smart Ecosystems [15]. The former is formed by software products. The latter is formed by Cyber-Physical Systems (CPS) and considers also hardware-software interactions in addition to the interaction between software components.

In the context of Smart Ecosystems, CESs reflect the system interaction and have relationships with other related entities like actors and goals as described in [26]. The functional and non-functional properties of a system reflect the goals of the actor(s) introducing the system on the market. A system, therefore, collaborates with another system according to the goals of the organization that is introducing it on the market. Due to difference of goals between organizations [32], systems engaged in a collaboration can expose different functionalities at different points in time. CESs can form collaborative system groups that function in the context of an ecosystem and influence the health of the ecosystem and its participants. The health of an ecosystem is a measure that describes the business opportunities of the ecosystem and its participants [40].

Automotive Smart Ecosystems [15] are a particular type of Smart Ecosystems. The CPS aggregated by an automotive smart ecosystem are vehicles. CPS are complex heterogeneous systems with a tight combination and interaction between embedded systems and physical elements [27]. A vehicle consists of embedded systems that collaborate with each other during intervehicular communication. Therefore, we consider vehicles as both CPS and CES. In the context of Automotive Smart Ecosystems, vehicles collaborate with each other in order to achieve business goals of several Original Equipment Manufacturers (OEM). For instance, platoons of collaborating vehicles can be formed with the objective of reducing fuel consumption. This use case is detailed next.

2.3 Vehicle platoon

A future-looking application of CES is formation of vehicle platoons on highways. The main idea is that vehicles form convoys or platoons traveling in the same direction by closely following each other. Vehicle platoons are dynamic in the sense that they can be formed and dissolved at any point based on the needs and goals of their participants, i.e. the individual vehicles. Vehicle platoons are expected to optimize traffic. They are also expected to have a positive effect on fuel consumption and CO2 emissions, since a vehicle-follower would take advantage of the reduced air friction when traveling close to the leading vehicles. Finally, vehicle platoons are expected to increase the safety in highways by advanced vehicle-to-vehicle communication and automation that, e.g., would allow complex maneuvering to prevent deadly accidents.

Fig. 1 depicts a CSG (vehicle platoon) formed by CESs that communicate with each other in context of an Automotive Smart Ecosystem. Actors (OEM) are part of automotive smart ecosystems. Actors provide, for example, software functions that enable formation of vehicle platoons within the ecosystem. Depending on the scope and context of forming a platoon, different rules for driving may apply. For example, an ad-hoc ecosystem formed at a construction place or a place of an accident can instrument the formation of a vehicle platoon with lower driving speed than an ecosystem instrumenting the formation of a vehicle platoon on highways.

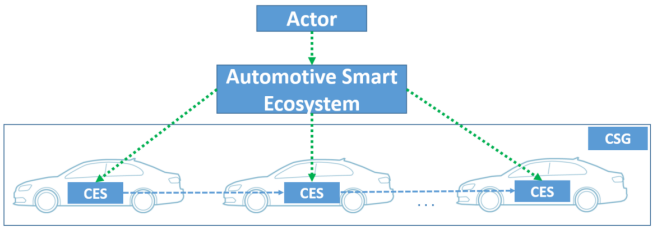

Fig. 1 CES and CSG in context of Automotive Smart Ecosystems

A vehicle platoon is a CSG since it aggregates individual CESs (the participating vehicles) which maintain their autonomy and communicate with each other via wireless channels. Each vehicle in the platoon is expected to be able to join and leave a platoon either from the front, the rear, or the side of the platoon. The functionality of "joining a platoon" is shared among different vehicles that need to coordinate. For instance, a vehicle that is joining a platoon from the rear may need to speed up in order to reach the platoon, while the platooon itself may slow down in order to facilitate the joining. More complicated scenarios include merging and splitting of existing platoons and dissolving of platoons.

A central challenge in developing vehicle platooning is the question of how to test this coordinated behaviour for the absence of faults, given that coordination happens at runtime based on some predefined logic that specifies when a vehicle can be part of a platoon, join a platoon, leave a platoon, etc. Current testing and simulation methods for automotive embedded systems assume that all functionality can be tested within the

system boundary of a single vehicle—a single CES [18]. In vehicle platoons, most functionalities to be tested extend the boundaries of a single vehicle. It becomes thus very challenging to apply existing procedures. Another challenge of testing and simulating vehicle platoon systems is that it typically involves different domains: (microscopic) traffic, (wireless) communication, and software. Hence, co-simulation techniques need to be employed.

2.4 Characteristics of Collaborative Embedded Systems

Schlingloff defines the term Embedded System (ES) as "a computational system, which is a fixed component of a technical system" [38]. ESs continuously process input and are capable to provide output in real time; their functionality is limited by the technical context in which they are operating. Depending on the programmed functionality and context of operation, ESs can supervise, control or acquire data in a fixed context.

Simulation tools and methods can be used for the evaluation of embedded systems in multiple contexts. Before looking on how evaluation of CES is performed, we introduce the key differences brought by behavioural evaluation of CES as compared to evaluation of single embedded systems and we conclude with a set of characteristics of CES that need to be verified and validated by simulation.

1. When systems meet and form collaborations at runtime, new behaviour can emerge. This behaviour needs evaluation through simulation and coupling to real world scenarios like real road traffic. Additionally, in order to facilitate reasoning about effects of emergent behaviour, visualization needs to be performed in an intuitive way.
2. Performing simulation-based evaluation of an embedded system implicitly takes into account the goals that the system needs to satisfy. In a CSG setting, multiple CES can have different, even conflicting goals. Therefore an important first step in evaluating their behaviour is to analyse each CES's goals and their contribution of the overall ecosystem goal (e.g. forming a platoon). So, simulation methods and tools should inherently support the modeling and analysis of system's goals and their connection to simulation outcomes.
3. In a collaboration, an embedded system becomes aware of information about the environment through exchange of functions and data. For example, in a vehicle an embedded system running a control function that processes information about the environment will make its decisions based on information received from other embedded systems located in another vehicle. The function of sensing the environment is passed to a collaborator through a set of messages. This opens up a wide range of uncertainties regarding the accuracy and timeliness of the information sensed, interpreted and delivered by a remote vehicle. Therefore, in evaluating a collaboration, uncertainties together with their effects need to be simulated. One way of doing this is by injecting faults like delayed communication.
4. Since CSG are open-ended and dynamic systems (the number and potentially collaborators or type of collaborators may change over time as the system runs) it is very difficult to test for all the different scenarios which might arise at runtime. An alternative to only design-time simulation is to use simulation methods and methods after deployment of a system as a means of runtime validation. As a result, simulation methods for CSG would need to be usable also at runtime.
5. Technically, Embedded Systems forming a collaboration can communicate over the same shared memory, over bus connection or over the Internet by means of 5G wireless communication [31]. Regarding the method of communication, ESs can communicate with each other in an event-based manner which makes an efficient use of network resources. This means that the processing of information is triggered by actions received from user input, alarms, or data inputs from other systems. In order to perform evaluation of CES and CSG, simulation of the communication channels needs to be performed.
6. Embedded systems function interact with the physical environment in order to fulfill the goals of the system. This implies that at the level of system analysis, simulation of the physical environment needs to be considered during evaluation of CESs.
7. Evaluation of CES and CSG need coupling of simulation models independently developed. The simulation models can be subsystems or systems communicating with each other or can be simulations of network communication. Co-simulation platforms are needed in order to evaluate emergent system behaviour.

All above mentioned points indicate the complexity of simulating embedded systems that collaborate with each other in order to provide required services to other systems or to the end-users. In what follows we present the methodology we've used to find answers to the challenges mentioned in this section.

3 Research Method

To provide an overview of simulation methods and tools for CES, we performed a review of literature. We were in particular looking for answers on:

- **RQ1:** What are the application challenges in the field of simulation for ES and CES?
- **RQ2:** In which phases of the software engineering lifecycle of ES and CES are simulation methods mostly used?
- **RQ3:** What is the scope of simulation methods in each phase of the software engineering lifecycle?

We performed a combined machine and human reasoning research strategy with tool support. During the research process, we identified papers that fall in four main categories, namely:

1. **Software Engineering**. Here, simulation methods and tools are mainly used for *System Modelling*, *Analysis of System Performance* and *What-If Analysis*, among others.
2. **Hardware or System Simulations**. Here, simulation methods and tools are used for, e.g. for simulating *Processor Architectures*, *Memories* or *Network on Chips*.
3. **Mathematical Simulations**. This more specific category refers to simulations used for approximating mathematical solutions (e.g. numerical solvers, Monte Carlo experiments).
4. **Simulations of Physical Phenomena**. In this case, simulations are used as a way to understand complex real-life phenomena such as earthquakes, dynamics of fluids, and biomolecular processes.

Since we are dealing with simulation as means of V&V in embedded systems, we restricted ourselves to publications belonging to the first category only.

Threats to Validity. We note that a main threat of our review is that it is not complete. The reason for this is related to the choice of search terms. We note that, since simulation is many times mentioned and used a by-product (albeit an important one) of other research approaches that are relevant to CES. For instance, CES can be conveniently modeled as agents, and as such, approaches from multi-agent research become relevant. Similarly, CSG can be seen as systems-of-systems, hence, literature in this line of research is also worth investigating in terms of simulation approaches it offers.

4 Simulation Methods and Tools for CES

In this section we present our results regarding identified methods and tools that can be used for VV activities of CES. Then, in Section 5 we focus on methods and tools targeting the automotive domain in particular. The identified general simulation frameworks are used during different phases of system development. Our survey shows, on the one hand, the diverse tool landscape and, on the other hand, that many challenges are already under active research.

In some studies, like the one presented in [27], simulating CES with virtual prototyping requires early execution of software binary code on target processor. This is necessary in order to identify compiler-specific effects on the target platform together with the impact to the overall behaviour of the system. Since very specific simulation models decrease the simulation speed considerably and make the whole V&V process infeasible, the underlying bus communication and middleware is abstracted depending on the scope of the evaluation. Another aspect of CES development is the modeling of communication. For this, in the study presented in [19], the Universal Simulation Engine [20] is used to exchange messages within a simulation.

In more detailed simulations, the close integration of embedded software with the physical processes of the system is achievable for single embedded systems. Thereby, effective virtual prototyping of embedded software can be realized in a mixed HW/SW simulation environment. In the study presented in [27], the authors present an embedded SW simulation with focus on the refinements of hardware-dependent software, realized with a toolset that integrates open tools and libraries based on C/C++. In addition, Open Dynamics Engine (ODE) [5] can be used for simulating physical rigid body movement and dynamics. SystemC/SystemC-AMS [6] is used for simulating mixed digital/ analogues systems together with the open source software emulator QEMU [7] that provides instruction and register accurate CPU abstraction of multiple instruction set platforms and is used for fast software emulation. Based on this, a simulation environment is used in [22] which can also embed additional simulators in addition to SystemC/SystemC-AMS. All these tools provide open interfaces to Matlab/Simulink [4] where additional design flow of physical components can be modeled.

Already developed methods and tools provide the basis for defining embedded systems at different levels of abstraction. On top of this, co-simulation methods are needed. On the one hand, it is clear from this summary that there is still no uniform procedure for the simulation-based V&V of CES. On the other hand,

so far only known methods are transferred to the new problems of collaborative systems.

In what follows we present the concept of co-simulation and how different tools can be used for co-simulating behaviour of automotive smart ecosystems.

4.1 Co-simulation

Simulation tools provide interfaces that enable the definition of simulation models, engines that perform the execution of the simulation models and means for visualizing the simulation results of systems and subsystems that share the same semantic domain. Various tools for simulation are used differently depending of the scope of simulation. Virtual evaluation of systems and component interaction requires composition of different semantic domains. Through co-simulation, system and system components modeled in continuous or discrete domains are put together to work. The Functional Mockup Interface (FMI) is the standard used for exchanging dynamic models and for improving their interoperability and re-usability. It enables exchange of models and coupling of simulators. In this context, High level Architecture Frameworks (HLA) provide a master algorithm that allows co-simulation by coupling simulation models defined at a high level of abstraction.

Co-simulation bridges not only domain-specific components but also tools used to model and simulate a system or a collaborative system group. Co-simulation is used for identifying problems in later stages of system development, that have not been discovered in the initial conceptual stage. It requires an integrating platform that enables coupling of different simulation models that are solved independently by domain specific simulators. Such platforms, like the one presented in [24], support exchange of intermediate results during the execution of the simulation models. Early testing of safety concepts that mitigate the impact of hardware failures on software behaviour is presented in [14].

In order to achieve co-simulation, the integration of different types of heterogeneous models is important. In the development of cyber-physical systems, UML and Matlab/Simulink are the most popular model types. The authors of [44] provide a method to link these two together based on a defined mapping and a synchronization of the time management that enables the simultaneous execution of simulation models.

Co-simulation of CESs requires coupling of simulation models of different processing and network platforms. Most often, communication networks are event-driven whereas processing platforms operate in the continuous time domain [17]. When an event in one domain occurs, information is passed to the other domain where further processing is performed. Co-simulation platforms have the scope of synchronizing both domains.

As summarized in [12], different methods for synchronization exist:

- Offline. For instance, a computational model is exported to C code, compiled, and imported to a network simulator for co-simulation.
- Master-Slave. One simulator is coordinating the co-simulation steps.
- Time-Step method. Simulators independently execute the behavioural models and pause at fixed interval of time in order to exchange information.
- Global Event-Driven. A global event list is used to keep a mixture of events received from the simulation models according to their timestamps.

In terms of scalability, the interaction of different simulators increases the volume of messages within the simulation environment. This can considerably slow down the overall performance. Especially with the integration of hardware, the simulators are often supplied with the same sensor data. In the approach presented in [19], the communication is optimized and reduced. Although it focused on distributed embedded systems, the approach can be extended to simulation of CESs.

5 Simulation of CES in the Context of Automotive Smart Ecosystems

5.1 Co-simulation for Automotive Smart Ecosystems

In automotive system development, the vehicle design follows the classical V-model [21]. The specification phase is followed by initial design of different components. As development advances, the simulation models are integrated and tested. This requires integration of different languages and tools from multiple domains. During these activities it is common to use approaches and toolsets based on Matlab/Simulink design flow. A simulation framework that supports integration of different simulation models is necessary from the early stages of system development. Tools like Virtual System Platform [9], Vista Virtual Prototyping [10] and Synopsys - Virtualizer [8] support Electronic Device Automation (EDA) through virtual prototyping for embedded software development in multi-language simulation environments. The FERAL [24] simulation platform provides the possibility to simulate control functions in interaction with virtual and real physical environment [15].

In the context of virtual evaluation of the formation and behaviour of a vehicle platooning, co-simulation

needs to be used in two operational modes. Cooperation of vehicles that join or leave a platoon needs to be simulated in the discrete-time domain while the behaviour of a single vehicle needs to be triggered by events. The events involved in creating, joining and leaving the platoon, for example, can be triggered by transitions in state charts. A high level of abstraction for behavioural description is suitable for analyzing the effects of decisions. Continuous behaviour of the platoon given by the driving behaviour is simulated in the continuous-time domain mostly by coupling Matlab/Simulink models. Collaboration between vehicles is achieved by the control algorithm of the Collaborative Adaptive Cruise Controller (CACCs) responsible for exchanging information with other CACCs.

When simulating a vehicle platooning, the vehicle, the network, traffic and the assistance system itself must be simulated. Rarely can a single simulation environment cover everything. One possibility of performing co-simulation in automotive smart ecosystems is by using the V2X Simulation Runtime Infrastructure (VSimRTI) [39] that has the ability to associate a traffic simulation such as SUMO [25] with a network simulation such as ns-3 [35] or OMNeT++ [42]. The exchange is realized via HLA and enables an extensible simulation of complex systems. In addition, vehicle models or other assistance systems can be added. The modular design enables a customizable environment for versatile use, and enables testing of collaborative systems. The networking of the systems is an essential part here. Other frameworks such as the vehicular network simulation framework Veins [41] support similar transmission possibilities. The versatile user interface shows the user what happens during a simulation run. On the other hand, there are no interfaces to simulators outside the network domain. Raith et al. [34] have also presented a framework for the test levels MiL (Model in the Loop), SiL (Software in the loop), HiL (hardware in the loop). The interesting point here is the networking of systems within a vehicle. This allows active (e.g. brake assist systems) and passive (e.g. airbag) assistance systems to communicate with each other, forming an integrated safety system. The approach was also implemented in a vehicleMaker® simulation [2]. The program simulates complete scenarios with different vehicle types. A connection to Matlab/Simulink or via FMI is also possible. For realism, real maps that are also used in navigation devices can be imported. A disadvantage is the idealized behaviour of the driver.

The approaches and tools presented in this subsection show that there is support for performing co-simulation in automotive domain for single systems and for collaborative systems even considering the human behaviour. HiL testing can be orchestrated with simulation tools and can therefore, be extended with pure simulated entities.

5.2 Integration of real world entities and virtual entities

In order to achieve a higher coverage during testing, there is the possibility of enhancing test scenarios formed by physical embedded systems communicating with each other. For example, communication with virtual systems can be integrated. In this way, an extended set of collaborations can be tested for a real physical system. A prototype platform that provides mechanism for coupling virtual systems with real physical systems to test CES groups in context of automotive domain is presented in [15]. The authors propose a modular approach of a testing platform that enables visualization of simulation results using gaming engines.

In [13], the authors introduce another approach for combining the advantages of driving simulators and real test vehicles by combining them in the traffic simulator provided by the company Vires Simulationtechnologie GmbH. Visualization of results is performed using Augmented Reality. Even though the platform presented offers possibility for testing single systems, it can be enhanced for testing ADASs (Advanced Driver Assistant Systems) that work cooperatively.

In a variety of possible scenarios, there are several approaches to select a suitable scenario for testing an ADAS. In [36], scenario definitions and real driving data are used to determine suitable scenarios for ADAS functions. Thus, scenarios can be identified based on criteria regarding temporal and spatial dependencies of road users. The “effect size” is used as a statistical indicator for autonomous driving. The method does not replace manual selection of scenarios, but provides guidance on the usability of the test environments.

An approach for testing single systems using virtualization is presented in [29]. Vision-based ADAS can be tested using visualization engines. Images from visualization tools can be fed into vehicle detection algorithms. This approach eliminates the risks for the driver and the vehicle during real test drive and provides reproducible test conditions.

Virtualization is further on used on testing human behaviour in single systems. The authors in [27] provide an approach for modeling the virtual body of the driver, divided into head, torso, arms and legs in order to achieve a correct tilting movement of the driver.

A method for translating real test drive into a simulation environment is presented in [28]. Real world driving behaviour is captured with a camera and used to

create real scenarios for the simulation environment. In particular, the course of the road, trajectory of the vehicle, trajectory and attributes of other relevant traffic, and information about the environment can be extracted. Weather parameters can also be included. Real driving can be reproduced in this way, but is not dynamically adaptable.

5.3 Summary of identified tools

In Table 1, Table 2 and Table 3 we summarize the set of tools identified when surveying the literature for answering our research questions (Section 3), together with how they address the challenges identified in Subsection 2.4. We acknowledge that this is not an extensive list of simulation tools focusing on the automotive domain. The entries in the *Co-Simulation* column reflect the capacity of the tool to support co-simulation by coupling different simulation models. *Evaluation of combined behaviour* reflects the capacity of tools to model and simulate multiple systems. Generally, even if a tool does not provide support for co-simulation, it can still provide support for evaluating combined behaviour of systems and system groups. *Visualization* relates to the capacity of tools to provide visualization of effects in an intuitive way. *Fault Injection* comprises the capacity of tools to inject faults in order to improve the test coverage of a scenario execution. *Simulation of communication channel* refers to the capacity of a tool to simulate communication channels like Wireless or CAN bus. *Simulation of physical environment* refers to the tool's support for simulating objects outside the system boundary with which the system interacts. *Used for V&V* summarizes the tool's capacity to be used during validation and verification activities, after the development process was completed. *Prototyping* summarizes the tool support for creating abstract simulation models for supporting the initial phases of concept exploration. *Simulation Type* refers to the simulation model supported by the tool. Finally, *Business model* classifies a tool as commercial or open source.

6 Simulation Challenges in the Context of Automotive Smart Ecosystems

Within a CES, embedded software interacts closely with the hardware layer. In the simulation methods and tools that have been discussed, the hardware platforms have been abstracted in order to achieve efficiency and make the simulation methods feasible for the scope of evaluation. The CESs are closely coupled with the technical platform on which they are executed. Compiler optimizations can influence the functionality delivered to a CES. Therefore, methods that accurately verify the function interaction between the hardware and the software layer for CESs are needed.

In the context of evaluating control decisions of embedded systems, research needs to be oriented towards developing simulation methods that enable execution of simulated behaviour of components at a much higher speed. These simulation methods need to take into consideration the characteristics of the hardware and software components.

The structure of a simulation depends largely on the system under test. Parallel to the development of the systems, the simulation environments have to be further developed. Due to the very high complexity of CESs, simulation scenarios are also highly complex. The evaluation of these scenarios is still a big challenge today. Often only specific incidents can be identified (e.g., accidents, collisions, message losses). However, the fulfillment of different requirements must be checked. Simulation results have to be better linked to the requirements. First approaches are discussed in [33]. Here, the developer has a lightweight approach available to check language requirements in simulations. In the future, more models and requirement types would have to be supported. Further research is needed to find automated approaches that handle the selection and prioritization of requirements and corresponding scenarios. In the automotive context, the creation and selection of simulation scenarios predominantly takes place manually. The first approaches to reducing the time required are the selection of critical scenarios. Another option is the custom creation of a scenario for the system. In both cases, it cannot be guaranteed that the scenario covers all the aspects of the system. Future automated approaches that extract such properties from the requirements can result in faster test cycles and better test coverage. The scenarios could be then created dynamically, depending on the system under test.

The degree of abstracting the reality is another challenge in simulations, especially in the automotive domain. While many approaches are concerned with the detailed modeling of the systems, there is still a big gap to e.g. realistically embed the driver's behaviour into a simulation. Especially the changeable unpredictable behaviour while driving and the operation of the ADAS are difficult to simulate with today's methods. This requires further research.

Simulation offers the possibility of testing collaborative systems in an extended set of contexts and in multiple collaborations. Further research and development activities that address the possibility of linking

Table 1 Overview of tools - Part 1

Tool	Co-Simulation	Evaluation of combined system behaviour	Visualization
FERAL [24]	Yes	Yes, through co-simulation	Yes, through coupling to Unity 3D
CarMaker [2]	No	No	Yes, through IPGMovie the virtual vehicle environment can be visualized
Virtual System Platform [9]	No	No	No
Vista Virtual Prototyping [10]	No	No	No
Synopsys Virtualizer [8]	No	No	No
Matlab/Simulink [4]	Yes	No	Only through plotting
VSimRTI [39]	Yes	Yes	No
SUMO [25]	No	Yes	Yes
OMNET++ [42]	No	Yes	Yes
Veins [41]	Yes	Yes	Yes

Table 2 Overview of tools - Part 2

Tool	Fault Injection	Simulation of communication channels	Simulation of physical environment
FERAL	Yes	Only at an abstract level	Only at an abstract level
CarMaker	Yes	No	Yes
Virtual System Platform	Yes	No	No
Vista Virtual Prototyping	Yes	No	No
Synopsys Virtualizer	Yes	No	No
Matlab/Simulink	No	No	No
VSimRTI	No	Yes	No
SUMO	street closures	No	Yes
OMNET++	network failures	Yes	Yes
Veins	street closures, network failures	Yes	Yes

Table 3 Overview of tools - Part 3

Tool	Used for V&V	Prototyping	Simulation Type	Business Model
FERAL	Yes	Yes	event-based time-based	Commercial
CarMaker	Yes, supports HIL testing	Not by using abstract simulation models	event-based	Commercial
Virtual System Platform	Yes	No	event-based time-based	Commercial
Vista Virtual Prototyping	Yes, by providing a hardware abstraction to software	No	event-based	Commercial
Synopsys Virtualizer	No	Yes, by linking virtual prototyping with FPGA-based prototypes	event-based	Commercial
Matlab/Simulink	No	Yes	event-based time-based	Commercial
VSimRTI	No	Yes	event-based	Open Source
SUMO	No	Yes	discrete-time	Open Source
OMNET++	No	Yes	event-based	Open Source
Veins	No	Yes	event-based, discrete-time	Open Source

simulation of systems to virtual games can provide the base for creative definition of testing scenarios.

7 Conclusions and Challenges Ahead

In this paper, we reviewed the state of the art and practice in using simulations as a method for V&V of CES. We were particularly focusing on systems that are equipped with open interfaces and can form collaborations at runtime with other systems that are independently developed and deployed. Such scenarios occur, e.g. in the case of two or more vehicles that meet and form a vehicle platoon.

From our review, we can conclude that the simulation methods and tools landscape is quite diverse and

rich of features that can be used in simulating advanced collaborative scenarios. Indeed, many of the existing methods and tools offer advanced visualization capabilities and can be used for injecting faults during simulation and verify effects of uncertainty. They are also able to simulate not only the software and hardware, but also the network communication and the physical environment. Finally, there are existing tools (such as Veins) that already perform co-simulation by coupling together other simulators. Such tools can be used also in modeling and simulating realistic collaboration scenarios, especially in the automotive domain.

What we believe is still missing is (1) the connection of the simulated entities with high level goals and requirements, and (2) the use of simulators (or the creation of new simulators) for runtime use. With respect to (1), we note that as the complexity of simulated scenarios increases (and simulating advanced CES scenarios certainly features high complexity) so does the importance of providing a high-level view on what is simulated and why. They "why" part could be answered if simulation methods incorporated a notion of requirement that is being checked via a simulation and a notion of the goals each actor has in a collaboration formed at runtime. This way, the simulation envelope would be enlarged and include actors-systems that feature different goals (including e.g. malicious ones).

With respect to (2), we believe that simulation methods will need to transcend the traditional boundary of offline operation and be used by the collaborative systems at runtime. One can argue of course whether the process of playing out different scenarios at runtime to decide on how to act can still be called simulation or should be called runtime decision-making and verification instead. Nonetheless, we believe that methods currently used for performing simulations and even tools currently used and operated offline by humans should be used by CES because the sheer number of the different scenarios that need to be covered by simulation-based evaluation is too large to be covered at design-time alone.

8 Acknowledgement

This work has been funded by the German Ministry of Education and Research (BMBF) through the BMBF project CrESt (Collaborative Embedded Systems).

References

1. Collaborative Embedded Systems (CrESt) project: https://crest.in.tum.de/
2. Ipg automotive gmbh, karlsruhe, germany. www.ipg.de, carmaker 3.5 (2011)
3. ISO 26262 -international standard for road vehicles-functional safety, http://www.iso.org/, 2011.
4. Matlab7simulink. https://www.mathworks.com/
5. Open Dynamics Engine. http://www.ode.org/
6. Open SystemC - tutorial. http://www.asic-world.com/systemc/tutorial.html
7. QEMU - homepage : https://www.qemu.org/
8. Synopsis - Virtualizer. https://www.synopsys.com/verification/virtual-prototyping/virtualizer.html
9. Virtual System Platform. https://www.cadence.com/content/cadence-www/global/en$_US$/home/tools/system−design−and−verification/software−driven−verification/virtual−system − platform.html
10. Vista Virtual Prototyping. https://www.mentor.com/esl/vista/virtual-prototyping
11. Bach, J., Bauer, K.L., Holzäpfel, M., Hillenbrand, M., Sax, E.: Control based driving assistant functions test using recorded in field data. In: 7. Tagung Fahrerassistenzsysteme (2015)
12. Besanger, Y., Tran, Q.T., Boudinnet, C., Nguyen, T.L., Brandl, R., Strasser, T.I., et al.: Using power-hardware-in-the-loop experiments together with co-simulation for the holistic validation of cyber-physical energy systems. In: Innovative Smart Grid Technologies Conference Europe, 2017 IEEE PES, pp. 1–6. IEEE (2017)
13. Bokc, T., Maurer, M., Farber, G.: Validation of the vehicle in the loop (vil); a milestone for the simulation of driver assistance systems. In: 2007 IEEE Intelligent Vehicles Symposium, pp. 612–617 (2007). DOI 10.1109/IVS.2007.4290183
14. Cioroaica, E., Jahić, J., Kuhn, T., Peper, C., Uecker, D., Dropmann, C., Munk, P., Rakshith, A., Thaden, E.: Accelerated simulated fault injection testing. In: Software Reliability Engineering Workshops (ISSREW), 2017 IEEE International Symposium on, pp. 228–233. IEEE (2017)
15. Cioroaica, E., Kuhn, T., Bauer, T.: Prototyping automotive smart ecosystems. In: 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W). IEEE (2018)
16. Cioroaica, E., Kuhn, T., Buhnova, B.: (Do not) trust in ecosystems. In: Proceedings of the 41th International Conference on Software Engineering (ICSE-NIER 2019). ACM (2019). Pre-print available at https://www.researchgate.net/publication/331498836_Do_Not_Trust_in_Ecosystems.
17. Cioroaica E. Kuhn, T.: Simulator coupling for network fault injection testing. In: Proceedings of EUROSIM Conference on Modeling and Simulation 2016. Linkping Electronic Conference Proceedings (2016). DOI 10.1109/EUROSIM.2016.160
18. Duracz, A., Eriksson, H., Bartha, F.A., Xu, F., Zeng, Y., Taha, W.: Using rigorous simulation to support iso 26262 hazard analysis and risk assessment. In: High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), pp. 1093–1096. IEEE (2015)
19. Fu, D., Becker, M., Szczerbicka, H.: On the potential of optimized data exchange in distributed embedded simulation. In: 2015 IEEE/ACM 19th International Symposium on Distributed Simulation and Real Time Applications (DS-RT), pp. 179–186 (2015)
20. Fu, D., Becker, M., Szczerbicka, H.: Universal simulation engine (use): a model-independent library for discrete event simulation. In: SpringSim (2015)

21. Hermans, T., Ramaekers, P., Denil, J., De Meulenaere, P., Anthonis, J.: Incorporation of autosar in an embedded systems development process: A case study. In: Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on, pp. 247–250. IEEE (2011)
22. Ilieskou, N., Blom, M., Somers, L., Reniers, M., Basten, T.: Multi-domain virtual prototyping in a systemc sil framework: A heating system case study. In: 2015 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), pp. 286–294 (2015)
23. Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L.: Recent development and applications of sumo-simulation of urban mobility. International Journal On Advances in Systems and Measurements **5**(3&4) (2012)
24. Kuhn, T., Forster, T., Braun, T., Gotzhein, R.: Feral-framework for simulator coupling on requirements and architecture level. In: Formal Methods and Models for Codesign (MEMOCODE), 2013 Eleventh IEEE/ACM International Conference on, pp. 11–22. IEEE (2013)
25. Lopez, P.A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Fltterd, Y., Hilbrich, R., Lcken, L., Rummel, J., Wagner, P., WieBner, E.: Microscopic Traffic Simulation using SUMO. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pp. 2575–2582 (2018). DOI 10.1109/ITSC.2018.8569938
26. Manikas, K.: Revisiting software ecosystems research: A longitudinal literature study. Journal of Systems and Software **117**, 84–103 (2016)
27. Mueller, W., Becker, M., Elfeky, A., DiPasquale, A.: Virtual prototyping of cyber-physical systems. In: Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific, pp. 219–226. IEEE (2012)
28. Nentwig, M., Stamminger, M.: A method for the reproduction of vehicle test drives for the simulation based evaluation of image processing algorithms. In: 13th International IEEE Conference on Intelligent Transportation Systems, pp. 1307–1312 (2010)
29. Nentwig, M., Stamminger, M.: Hardware-in-the-loop testing of computer vision based driver assistance systems. In: 2011 IEEE Intelligent Vehicles Symposium (IV), pp. 339–344 (2011)
30. von Neumann-Cosel, K., Dupuis, M., Weiss, C.: Virtual test drive-provision of a consistent tool-set for [d, h, s, v]-in-the-loop. In: Proceedings of the Driving Simulation Conference Monaco (2009)
31. Popovski, P.: Ultra-reliable communication in 5g wireless systems. In: 1st International Conference on 5G for Ubiquitous Connectivity, pp. 146–151 (2014). DOI 10.4108/icst.5gu.2014.258154
32. Popp, K.M.: Goals of software vendors for partner ecosystems–a practitioner s view. In: International Conference of Software Business, pp. 181–186. Springer (2010)
33. Pudlitz, F., Vogelsang, A.: A lightweight multilevel markup language for connecting software requirements and simulations. In: 25th Intl. Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ) (2019)
34. Raith, A., Sattler, K., Ertlmeier, R., Brandmeier, T.: Networking and integration of active and passive safety systems. In: 2011 Proceedings of the Ninth International Workshop on Intelligent Solutions in Embedded Systems, pp. 75–80 (2011)
35. Riley, G.F., Henderson, T.R.: The ns-3 Network Simulator. In: K. Wehrle, M. Gne, J. Gross (eds.) Modeling and Tools for Network Simulation, pp. 15–34. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
36. Roesener, C., Fahrenkrog, F., Uhlig, A., Eckstein, L.: A scenario-based assessment approach for automated driving by using time series classification of human-driving behaviour. In: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), pp. 1360–1365 (2016)
37. Rosen, R., Von Wichert, G., Lo, G., Bettenhausen, K.D.: About the importance of autonomy and digital twins for the future of manufacturing. IFAC-PapersOnLine **48**(3), 567–572 (2015)
38. Schlingloff, B.H.: Cyber-physical systems engineering. In: Engineering Trustworthy Software Systems, pp. 256–289. Springer (2016)
39. Schnemann, B.: V2x simulation runtime infrastructure vsimrti: An assessment tool to design smart traffic management systems. Computer Networks **55**(14), 3189 – 3198 (2011)
40. da Silva Amorim, S., Neto, F.S.S., McGregor, J.D., de Almeida, E.S., von Flach G Chavez, C.: How has the health of software ecosystems been evaluated?: A systematic review. In: Proceedings of the 31st Brazilian Symposium on Software Engineering, pp. 14–23. ACM (2017)
41. Sommer, C., German, R., Dressler, F.: Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. IEEE Transactions on Mobile Computing **10**(1), 3–15 (2011). DOI 10.1109/TMC.2010.133
42. Varga, A., Hornig, R.: An overview of the OMNeT++ simulation environment. p. 60. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) (2008). URL http://dl.acm.org/citation.cfm?id=1416222.1416290
43. Wachenfeld, W., Winner, H.: The Release of Autonomous Vehicles, pp. 425–449. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
44. Wang, Y., Zhou, X., Liang, D.: Study on integrated modeling methods toward co-simulation of cyber-physical system. In: 2012 IEEE 14th International Conference on High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems, pp. 1736–1740 (2012). DOI 10.1109/HPCC.2012.261
45. Zofka, M.R., Kohlhaas, R., Schamm, T., Zollner, J.M.: Semivirtual simulations for the evaluation of vision-based adas. In: Intelligent Vehicles Symposium Proceedings, 2014 IEEE, pp. 121–126. IEEE (2014)