ATHENA PUBLISHING

Journal of Automotive Software Engineering

ISSN (Online): 2589-2258 **ISSN (Print):** N/A

Journal Home: https://www.athena-publishing.com/journals/jasen

Article Title

Multi-Level Message Sequence Charts to Validate the Collaborative Automotive Cyber-Physical Systems

Authors

Marian Daun, Bastian Tenbergen, Jennifer Brings, Patricia Aluko Obe

Corresponding Author

Marian Daun – marian.daun@paluno.uni-due.de

Cite This Article As

M. Daun, B. Tenbergen, J. Brings, P. Aluko Obe. Multi-Level Message Sequence Charts to Validate the Collaborative Automotive Cyber-Physical Systems. Journal of Automotive Software Engineering, Vol. 2(1), pp. 27–45, 2021.

Link to This Article (DOI)

https://doi.org/10.2991/jase.d.210710.001

Published on Athena Publishing Platform

31 January 2022

Journal of Automotive Software Engineering
Vol. **2(1)**, 2021, *pp.* 27–45
DOI: https://doi.org/10.2991/jase.d.210710.001; ISSN: 2589-2258
https://www.atlantis-press.com/journals/jase/

Multi-Level Message Sequence Charts to Validate the Collaborative Automotive Cyber-Physical Systems

Marian Daun[1,*], Bastian Tenbergen[2], Jennifer Brings[1], Patricia Aluko Obe[1]

[1] *University of Duisburg-Essen, Essen, Germany*
[2] *State University of New York, Oswego, New York, USA*

ARTICLE INFO

Article History

Received 02 Feb 2021
Accepted 11 May 2021

Keywords

Multi-level modeling
Cyber-physical Systems
Interaction scenarios
Message sequence charts
Validation
Defect detection

ABSTRACT

Autonomous driving and e-mobility are swiftly becoming not only the work of science fiction or popular science, but a reality. A key focus of manufacturers and suppliers in the automotive domain is of course to specify systems that implement this reality. Often, scenarios at type-level are used throughout the development process to specify system behavior and interaction within the car, as scenario models are comparatively easy to understand and can easily be subjected to manual validation. However, autonomous driving and e-mobility require interaction not just of systems within the same car, but collaboration between multiple cars as well as between cars and miscellaneous road infrastructure (e.g., smart road signs). The car becomes a Cyber-Physical System that dynamically forms collaborating networks at runtime with other Cyber-Physical System to create functionality that goes beyond the scope of the individual vehicle (e.g., resolve a traffic jam). Consequently, a plethora of possible compositions of such a network exist and must be specified and validated completely to assure their adequate and safe execution at runtime. Doing this at type-level with scenario models becomes prohibitively tedious, error prone, and likely results in unrealistic development cost. To combat this issue, we investigate the use of multi-level Message Sequence Charts to allow for specifying interaction scenarios between collaborative Cyber-Physical System in a network of collaborating automotive Cyber-Physical System. To assist the developer in systematically defining multi-level Message Sequence Charts, we propose two processes. The resulting diagrams use a mixture of type and instance-level abstractions within one conceptual diagram. This allows reducing the required effort to manually validate the adequacy of scenarios to a manageable amount because information within the scenarios can be validated in batches. At the same time, instance-level defects become more obvious. Evaluation results from a controlled experiment show that multi-level Message Sequence Charts contribute to effectiveness and efficiency of manual validation for collaborative automotive Cyber-Physical System.

1. INTRODUCTION

Autonomous driving and e-mobility seem to be the next milestone in the automotive industry [1]. This new territory comes with new challenges, particularly for software engineering [2]. For example, as cars roam traffic autonomously, they must interact with other systems such as other cars and smart traffic infrastructure to deal traffic situations that go beyond each individual car's ability to deal with, like the resolution of traffic jams through the interchange of distance and speed telemetry among cars with smart automotive cruise controls [3] (much like IoT systems [4]). In this sense, cars become Cyber-Physical Systems (CPSs), as they functionally collaborate with other systems around them (cf. [5]). To specify the functionality of automotive systems, often scenario models are used at the type-level to validate their correct and adequate behavior. Yet, as cars functionally collaborate with other automotive CPS, validating functionality is no longer a concern of the individual system, but concerns the collaborative network composed of several systems (see [5–8]). This is especially the case when functional inadequacies (in the following "defects") only occur in certain runtime interactions, e.g., when at least two systems of a certain type interact with one another (like in the traffic jam resolution example above). In such cases, there is a risk that defects may remain covert, as they are not displayed explicitly in a type-level model.

1.1. Validation of Collaborative CPS

A central aspect becomes that car's functional dependence on other external systems. For example, cyber-physical adaptive cruise controls [3] perform certain tasks independent from one another (e.g., selecting the speed at which to move the vehicle), but require the specification of behavior in collaboration with external systems (e.g., to navigate at a common velocity given other cars in the vicinity).

This interdependence must not only be (semi-automatically) verified (e.g., to check functional correctness [9]), but also manually

* *Corresponding author. Email: marian.daun@paluno.uni-due.de*

validated to ensure adequate functional [10, 11] adequacy.[1] In fact, many safety standards (e.g., [12–14]) require manual validation (through, e.g., reviews) of all design artifacts at certain points during development. For this purpose and depending on the OEM and system under development, Fagan inspections [15,16], walkthroughs [17], N-fold inspections [18], checklist-based inspections [19], or other manual validation techniques [20] are conducted to identify defects in requirements and development artifacts as early as possible.

However in case of such collaborative automotive CPS, inspecting the multitude of traffic scenarios (see, e.g., [21]) becomes a very tedious, time-consuming, and error-prone task. Nevertheless, it is essential to adequately consider all possible interaction scenarios that can occur at run time [22]. Interaction scenarios differ w.r.t. the number of collaborating cars, the different system types collaborating (car, smart road sign, traffic guidance system, etc.), or the surrounding systems in the context (how many cars there are in the traffic jam), etc. [23]

1.2. Need for Multi-Level Modeling

A typical solution to the multitude of concrete scenarios to be considered is the use of more abstract scenarios on type-level. However, the validator will detect some defects only when considering a concrete situation, which is not explicitly depicted in abstract type-level information [24]. For instance, there is a risk that the negotiated convoy velocity in a traffic jam is too high might only be recognizable in certain traffic situation (e.g., if the traffic jam occurred due to a construction site). Hence, in the case of automotive CPS, the interaction scenarios between CPS must be closely investigated. Yet, concrete interaction scenarios may comprise a vast amount of individual cars [21], external systems of different types, revisions, and vendors [23]. In consequence, the possible run time interactions can conceivably become infinite, such that the number of concrete instance models (depicting individual interaction scenarios) becomes (countably) infinite as well.

Hence, considering all possible interaction scenarios during the engineering of these systems is not realistic. Therefore, models are needed that allow investigating interaction scenarios on an instance-level, while at the same time limiting the number of scenarios to be investigated. This must be done in a way that accurately depicts inadequate interactions between system instances, without abstracting them away to type-level. Therefore, multi-level modeling provides a solution approach that allows to define models on a multi-level, i.e., defining some common aspects on the type-level to limit the number of models to be investigated, while defining interaction aspects on the instance-level that need to be investigated closely.

1.3. Contribution

This paper investigates the use of multi-level Message Sequence Charts (ml-MSC) to improve the validation of collaborative automotive CPS. We propose two processes to assist the developer in making conscious choices about which pieces of information to include from the type-level and from the instance-level. We propose rules for the structured development of ml-MSCs which do not purely consist of concrete system instances but also of abstract representations of common properties of comparable instance types. Purpose-specific abstractions are used to deliberately interrelate type-level and instance-level information within the same diagram. ml-MSC hence provide an intermediate abstraction between instance models and type models, which allows considering all possible run time interaction scenarios in a manageable fashion without losing information that hinders the detection of functional defects. We show their application through a running example of an adaptive cruise control system.

To investigate the benefit of this approach, we report on a controlled experiment. Results show that compared to type-level scenarios and instance-level scenarios the use of the proposed ml-MSC is advantageous in terms of expressiveness, effectiveness, and efficiency.

1.4. Outline

In the following, Section 2 discusses related approaches and related studies. We introduce the running example in Section 3. Section 4 introduces the foundations of ml-MSC, while Section 5 proposes structured approaches for the purpose-specific definition of ml-MSC and shows their application in the automotive domain. To aid adoption through tool support, Section 5 also briefly discusses the formal integration of ml-MSCs in relationship to ordinary MSCs [25]. Sections 6 and 7 discuss the experimental design and results of hypotheses tests, respectively, before Section 8 discusses the findings. Finally, Section 9 concludes the paper.

2. RELATED WORK

This section gives a brief overview over further related approaches from the state of the art. Particular emphasis is given to the distinction of type- and instance-level in multi-level modeling approaches as well as the specification of collaborative CPS, not necessarily limited to the automotive domain. In addition, we discuss similar studies concerned with validation and verification of engineering artifacts.

2.1. Multi-Level Modeling and Abstraction on Type- and Instance-Level

The principle idea behind multi-level modeling is to be able to retrieve type- and instance-level information individually and independently from one another, such that the abstraction can be dynamically chosen depending on the purpose of the model. Using scenarios, this was achieved in [26] by applying polymorphic scenario-based specification models in order to implement interfaces within instances. However, this requires sophisticated and unambiguous instance-level and type-level representations (see [27,28]). While this approach can be considered a multi-level modeling paradigm, the ability to abstractly represent common properties of comparable instance types remains desirable [29–32].

In [33,34], a different definition for multi-level modeling is proposed, such as the meta-language defines the context by means of variability indicators from [34]. The benefit of this approach

[1] In the following, we adopt Glinz' [11,10] differentiation between a system's ability to operate provably *correct* and a system's functionally *adequate* behavior given an operational scenario.

is that both ontological relationships (beyond the interface–implementation relationship from [26]) as well as classification of metaphysical structures [35,36] can be represented. This allows for modeling relationships between, but also within the same level of abstraction.

The levels of abstraction need not necessarily be always precisely two [37–39]. Instead, as shown in [40], classic transformation languages can be used to interrelate models on several abstraction levels to support multi-level modeling. A remaining challenge then becomes to guarantee consistency between interrelated models, which has been addressed, e.g., in [41]. Yet, while these approaches are ontologically grounded, the manual effort involved in creating multi-level models must not be ignored, but must be compatible with contemporary requirements and modeling techniques [42]. Since this effort gives rise to issues beyond those of consistency (as addressed in [41]), an approach to automatically analyze predefined integrity constraints in multi-level models is presented in [43].

Despite all these advances and benefits of multi-level modeling, one remaining central challenge is the increasing complexity within the models. The work we present in the following can be used to address this issue. In order to get an accurate overview of similarities and differences, the contribution of Igamberdiev *et al.* [38] provides a comparison focusing on language technology, domain modeling, and tool support. Similarly, the work by Atkinson *et al.* [29] also compares modeling approaches and their multi-level modeling, while in [44] the authors consider the evaluation of internal and external qualities. Lastly, Macías *et al.* [45] present an approach which can be used to uncover conflicts due to collaborative modeling environments, e.g., [46,47].

2.2. Multi-Level Modeling for Collaborative CPS

To account for specific characteristics of collaborative CPS, specific approaches have been proposed. A framework for defining inconsistency patterns and their respective management alternatives is presented in [48]. The approach is concerned with optimizing the modeling process with regard to various optimization criteria such as consistency and costs.

Besides the consistency validation, it is essential to validate the design during software and hardware configuration of CPS [5]. Since such configurations are not automatically feasible, Nie *et al.* [49] provides guidelines to simplify the process.

Another approach involving self-adaptive workflows in CPS is offered in [50]. Similarly, in order to gain an overview of the complexity of CPS, multi-paradigms for modeling in [51] are proposed, which intend to model each part of such complex systems explicitly. Not only their complexity poses a challenge for practitioners and researchers, but also the system integration which needs to be addressed to overcome these problems (see, e.g., [52–54]). In Mosterman and Zander [55], potential handling mechanisms for system integration in network infrastructures are identified and discussed for CPS is discussed, e.g., to improve security of interacting collaborative CPS across multiple levels of abstraction. Similarly in [56], regulations for collaboration are defined and protective mechanisms are determined that span multiple modeling levels.

2.3. Modeling During Validation and Verification

Our work is intended to primarily foster the manual validation of the functional adequacy of a system during early stages of development. This work complements our previous work on using diagrammatic representation to validate stakeholder intentions [57–59], safety properties [24,60,61], and context assumptions [62]. Our work has consistently shown that dedicated review models, specific to the purpose of validation, are more effective in uncovering defects in system requirements. For example, we compared ITU MSCs with other modeling languages as review artifact [57–59] showing that MSCs are a good review model. In addition, we have shown that MSC-based review models support both less experienced and more experienced reviewers [63] that instance-level MSC are a significantly better review artifact for collaborative CPS than type-level MSC [64]. However, as the problem of large sets of slightly differing instance-level MSC to be investigated remains, in this paper, we investigate whether the use of ml-MSC can provide a solution.

Outside of these efforts, only little work is available on manual validation of system properties. Miller *et al.* [65] report on an experiment with trained student participants, finding that perspective-based reviews are more effective than checklist-based approaches for error detection in natural language requirements specifications. Basili *et al.* 1996 [66] report on a controlled experiment with professional software developers. They conclude that the perspective-based review is significantly more effective than other inspection techniques for requirements documents. In replications various replications and comparable experiments these findings have been supported [67–73].

Most approaches concerned with quality assurance of engineering artifacts focus on functional correctness and the formal verification or automatic analysis thereof, as opposed to the validation of functional adequacy (see [10,11] for a discussion of the difference). A plethora of approaches deal exclusively with formally verifying dynamic properties of systems. A comprehensive thereon with specific focus on autonomous robotic systems is provided in [74]. Of all these approaches, particularly noteworthy are is the work by Brill, Damm, Klose, Wittke *et al.* [75–77] and by Bontemps *et al.* [78] on Live Sequence Charts. These are a extension to MSCs [25] to allow for verification of, e.g., partial ordering of messages or liveness properties, and through extensions, verification of time behavior [79,80].

A comparable extension to Petri Nets using Colored Petri Nets is proposed in [81]. This approach specifically aims to improve modeling of message transitions and token instantiations across different levels of abstraction, thereby enabling correctness checks of the system pertaining to different application domains. According to Frank [32], this may overcome some of the limitations impairing multi-level modeling at large, such as validation and verification across levels of abstraction.

Finally, Timed Automata [82] have been applied to multi-level modeling specifically to enable verification of safe states during autonomous vehicle maneuvers [83,84]. While less abstract than the work on Colored Petri Nets in that it applies to the automotive domain specifically (as opposed to the meta-domain approach in

[81]), the work by Hilscher, Schwammberger *et al.* [83,84] tackles safety defects of autonomous systems to the state space of the operational situation, rather than the interaction of functionally collaborating systems. In this sense, we consider the work by Hilscher, Schwammberger *et al.* [83,84] orthogonal, yet compatible to ours as it focuses on the same type of problem, yet from a different perspective (i.e., correctness of safe states as opposed to adequacy of functional collaboration).

3. RUNNING EXAMPLE

We illustrate the concept of ml-MSC throughout the remainder of the paper using a cooperative adaptive cruise control (CACC, e.g., [3]) case example.

Collaborative CPS dynamically form networks of collaborating CPS at run time in order to fulfill overall goals which cannot be fulfilled by the individual CPS independently. Such collaborative networks enable the individual systems to enhance their abilities using the functionality provided by other systems. In the case of a CACC, individual vehicles form a platoon at run time traveling with a common speed and decreased safety distances between one another. This allows the platoon itself to contribute to the reduction of emissions to the environment (due to reduced acceleration and deceleration and slipstream driving). Additionally, the traffic throughput on motorways is considerably increased [3,85]. Both goals cannot be achieved by any individual CACC, but only in collaboration among several CACCs.

Typical cruise control systems measure their "own" vehicle's current speed and control engine torque to maintain a constant speed desired by the driver. Adaptive cruise control systems enhance this functionality by also measuring the distance to vehicles driving ahead and actively apply the brakes to maintain distances between cars (avoid rear-end collisions). Cooperative adaptive cruise controls further enhance this further by allowing individual vehicles equipped with a CACC to form platoons by communicating with nearby vehicles. This is shown in Figure 1, where the desired speed (v) is transmitted and compared against the actual speed of the vehicle (v').

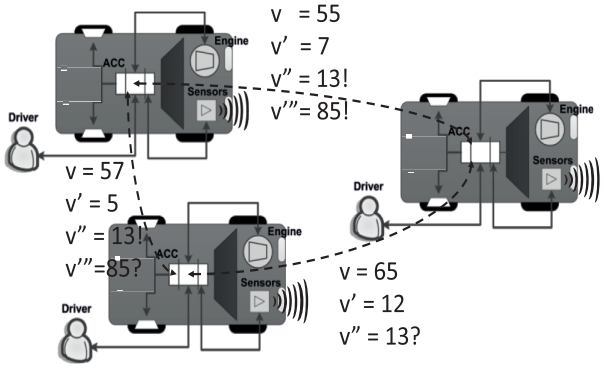

Figure 1 CACCs negotiate speed to form and coordinate platoons. "?" and "!" represent requested and agreed information, respectively.

Platoons form when at least two vehicles traveling into the same direction with comparable speed v' are within an appropriate range to one another. Platoons dissolve when the last two vehicles of the platoon depart. In between, other vehicles may join and leave the platoon. During operation the platoon negotiates a common speed between all partaking vehicles (v" in Figure 1). In a more advanced scenario, platoons conduct coordinated overtaking maneuvers and coordinated emergency brake maneuvers (if the lead vehicle detects an obstacle), and account for allowing other vehicles to cross the platoon (e.g., to allow that vehicle to move toward the highway exit ramp). For example, in Figure 1, the lower left vehicle requests (represented by "?") the upper left vehicle to overtake the vehicle in front of both vehicles (v'''). The upper left car responds by assuming a velocity of 85 ("!" in Figure 1) temporarily before coasting back down to the common speed v".

Because vehicles can join or leave the platoon, the platoon can consist of different numbers of vehicles (i.e., instances) at different points in time. For example, in order for one vehicle to overtake another, the platoon must be able to "make room" such that one vehicle can safely pass between gaps. Therefore, the platoon must also consider each vehicle's specific properties (e.g., physical dimensions, relative location within the lane) and functional abilities (e.g., maximum brake force, maximum acceleration, ability to partake in such a maneuver, etc.). The specific composition of the platoon at run time, hence, leads to a different behavior in building the gap.

4. FUNDAMENTALS OF MULTI-LEVEL MSCs

Model-based engineering is often centered around the use of graphical models [86]. Graphical models aid in the development of an architecture [87–89] as well as in interpreting, discussing, and negotiating design decisions [90]. Beside graphical diagrams, model-based approaches typically define more or less formal semantics of the models [91]. For the definition of purpose-specific views and abstractions that are integrated into one model, two international standards are often seen as the basis:

- ISO/IEC/IEEE 42010 [92] defines views and viewpoints for architecture descriptions to address stakeholder concerns.
- ISO/IEC 19508 [93] defines the Meta Object Facility (MOF) by the Object Management Group (OMG). The MOF provides a fundamental framework to define modeling languages that can be easily integrated with one another.

The MOF proposes the use of four meta layers [93]: a layer defining the commonalities between all modeling languages (meta-meta-layer), a layer defining the modeling language (meta-layer), a layer defining the model (model layer), and the layer defining the concrete real-world instances of the model (instance layer). Hence, in model-driven development, two model types are commonly differentiated:

- *Type-level models* specify the software or system to be developed on an abstract level. Hence, type-level models are useful to manage the combinatorial complexity resulting from the many possible combinations of the network of collaborating automotive CPS that must be accounted for [94]. However, the

risk remains that a certain defect is not detected during manual reviews due to the high-level of abstraction.

- *Instance-level models* highlight concrete scenarios [94], a concrete system of a specific type might face during run time execution. Therefore, they are a good means for stakeholder discussions and to detect errors and defects [95]. They are thus useful to inspect concrete functional inadequacies that are related to a certain composition of the network of CPS.

In the next subsections, we introduce ITU MSCs and show how the differentiation between type-level models and instance-level models manifests for these.

4.1. ITU Message Sequence Charts

ITU MSCs [25] are a formal diagram type that can be used to document interactions between the system and external actors as well as between system components. They are considered useful for specifying the interaction-based behavior of embedded systems [96] and are commonly proposed to specify scenarios [94].

There are two kinds of MSCs, high-level Message Sequence Charts (hMSCs) and basic Message Sequence Charts (bMSCs). hMSCs are directed graphs whose nodes reference other MSCs (either a bMSC or another hMSC). They can be used to structure the behavioral specification of the system. bMSCs specify the interaction between instances by means of messages. Specifically, bMSCs document the interaction between instances, the system, its components, or components of external actors. These are depicted as lifelines, which interact by exchanging messages. In the following, we hence focus on bMSCs exclusively.[2]

According to well-established formalizations (e.g. [97]), we define a single bMSC b as 5-tupel $b = (I_b, M_b, E_b, \leq_b, \alpha_b)$, where I_b is a set of lifelines, M_b a set of messages exchanged between the lifelines, and E_b a set of events (i.e., discrete points in time where one lifeline receives or sends one message). These elements are ordered using two relations: $\leq_b \subseteq E_b \times E_b$ defines the chronological order of occurrence of the events, $\alpha_b \subseteq E_b \times M_b \times I_b$ defines the relation between an event, a message and a lifeline. Figure 2 shows an exemplary bMSC and provides an overview of the basic graphical syntax of bMSCs.

[2] It is to note that the ITU standard [25] uses the term "instance" instead of "lifeline." However, we use the UML-common term "lifeline" to refer to bMSC instances to avoid confusion with system instances.

The eight bMSC lifelines defined in Figure 2 represent the platoon's eight cars. Sending and receiving of messages are considered events; in this example, the sending and receiving of messages with the cars' current speeds (v) and desired speeds (v'). Messages are causally ordered along the instances' lifelines, which means that, while messages are sent in the order that is depicted, they do not necessarily have to be received in the depicted order. Messages are sent in the depicted order after all previous incoming messages have been received. In this case, the lane switching maneuver is enacted. The lifeline for car "c3" requests "c5" to move closer to "c6" and "c7," ignoring safety distances. bMSC lifeline "c5" complies and informs "c4" that the maneuver is complete. bMSC lifeline "c4" then moves into the gap in the other lane.

4.2. bMSC Type-Level Modeling

For several reasons, model-based specifications are typically defined on a type-level. One key reason is that the abstraction provided by type-level models helps to cope with the complexity of specifications and to reduce their size [98–100]. In addition, type-level modeling is resembling the software implementation more closely, as software is also written on the type-level, despite being deployed to the instance-level. In the case of the CACC, a bMSC in a type-level model specifies system types, subsystem types, or component types as lifelines. These are the types typically specified in the system architecture (see Figure 3 for a simplified example). Typical lifelines would be:

- the CACC itself,
- CACC components (e.g., the distance sensor to detect other vehicles),
- human users interacting with the system (e.g., the driver, who steers the vehicle or sets the desired speed in the CACC),
- neighboring systems within the same car that the CACC interacts with (e.g., the engine controller or the brake controller to manipulate the vehicle's speed), and
- one single lifeline representing other CACC systems the CACC shall collaborate with during operation.

Figure 4 gives an example for such a type-level bMSC. As can be seen, the bMSC contains six lifelines on the type-level. Since

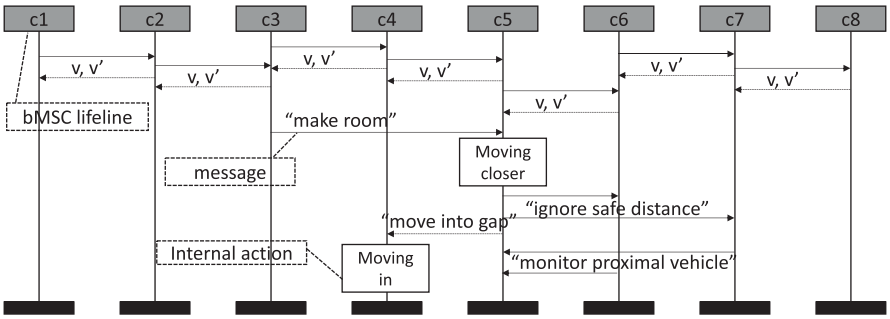

Figure 2 | A bMSC showing a lane switch maneuver. In this and the following figures, dark gray lifelines represent instance-level lifelines. Please note that color shading is not part of the ITU MSC standard [25] and was added by the authors to increase clarity.

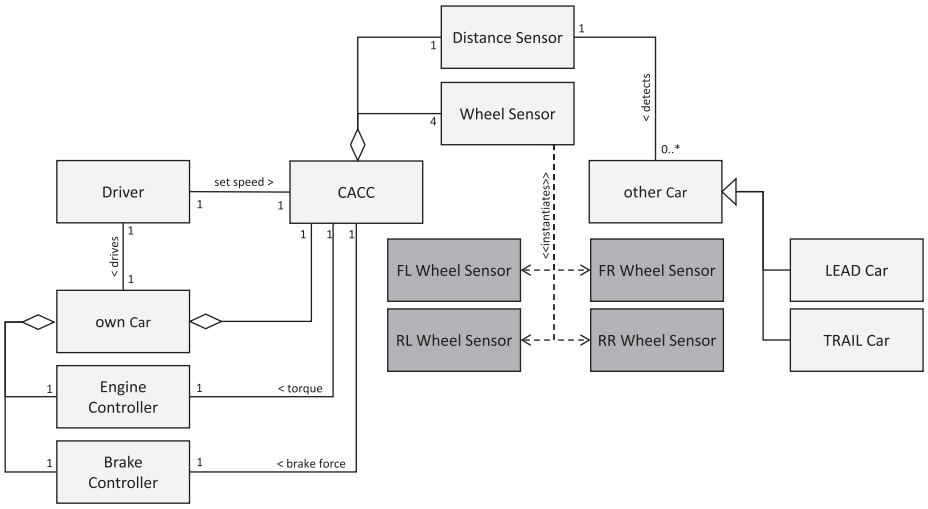

Figure 3 | Type-level architecture of the CACC. Type-level classes are depicted in light gray. Their instantiations are depicted in dark gray.

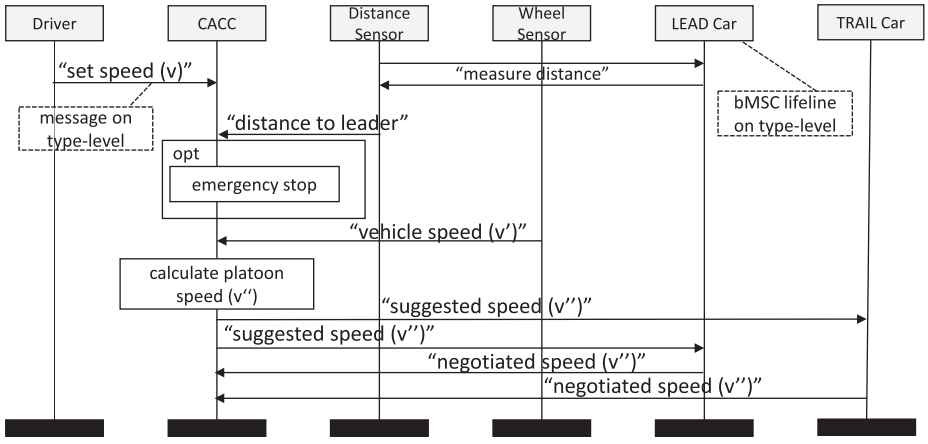

Figure 4 | Type-level bMSC for platoon speed negotiation. In this and the following figures, light grey lifelines represent type-level lifelines. Again, color shading was added by the authors to increase clarity.

these document types of systems rather than concrete instances, the exchanged messages are on the type-level as well. In this case, an excerpt of a simplified algorithm is shown, which documents how a CACC negotiates a platoon speed with the lead car and the trailing car. Based on the observed distance and desired driver speed, the CACC calculates an optimal platoon speed and submits this value to the leading car and following car. These cars contain CACCs themselves and hence submit a similar suggestion to the CACC of their own vehicle.

4.3. bMSC Instance-Level Modeling

On the instance-level, scenarios typically describe concrete situations [94], e.g.:

Trevor drives a black Vapid Motorcompany Fortune GT with a built-in CACC produced by ACME Inc. On the motorway Trevor's Vapid meets a Vulcar Nebular Turbo with a built-in CACC by AutoCorp. Both CACCs identify that they are driving in the same direction and hence establish a platoon.

Modeling approaches exist that transfer this concept of instance-level descriptions to a graphical model (e.g., [27,101,102]). Instance-level bMSCs describe a maneuver of the platoon on a concrete level for a concrete number of cars, drivers, and CACCs involved, as is shown in Figure 5.

In Figure 5, seven bMSC instance-level lifelines are shown. Note that in this case we do not necessarily refer to concrete instances but also groups of instances, i.e., some bMSC lifelines are rather on a instantiated type-level. While "Frank" is clearly an instance-level description, an "ACME CACC" is not on the abstract type-level of a generic CACC but has been instantiated to account for a specific vendor. However, it does not refer to just one concrete CACC build in a particular car. In Figure 5, the messages are also on the instance-level. In this case all messages clearly belong to the instance-level. The CACC by ACME Inc reads a concrete distance of 25m to the Vulcar equipped with the AutoCorp CACC. It also reads the Vapid's current speed of 14mph. Meanwhile, the AutoCorp CACC reads the distance to its neighboring vehicle (not depicted in Figure 5, however indicated by the ellipsis symbol "…" on the right) and its own vehicle's speed. The ACME CACC and AutoCorp CACC both

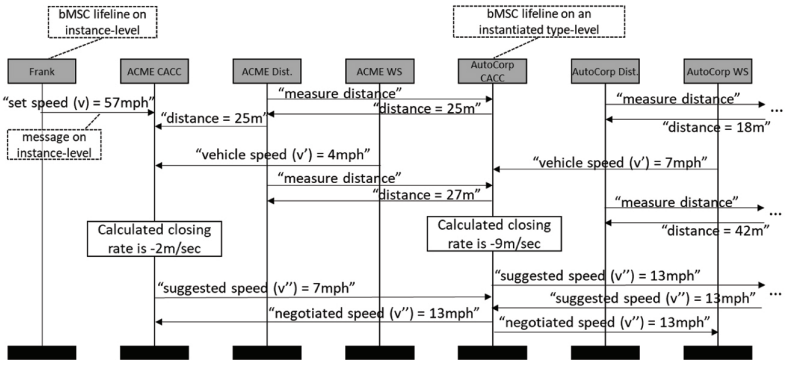

Figure 5 Instance-level bMSC for speed negotiation between CACCs. Like before, dark gray shading is used to signify instance-level lifelines.

initiate speed negotiations based on their respective calculated closing rate, but since the AutoCorp CACC measures a faster increase to its leading vehicle, the suggested speed of 17mph is increased to 19mph.

4.4. Type- Versus Instance-Level Validation

Type-level models allow investigating all situations abstractly and therefore aid validation. However, not all defects can be detected on the type-level, as they are abstracted away. Hence, instance-level models are needed. Another shortcoming of type-level specification lies in the support for elicitation of requirements, which is a strong argument for using scenarios in the first place [94]. Scenarios describe concrete detailed situations that are often specified on an instance-level, as it is hard to directly abstract from a real-world situation to a type-level specification without the intermediate step of writing down the original scenario on the instance-level [103]. Therefore, validates may find it easier to understand instance-level models during inspections or reviews.

Defects can result from the interplay of several instances of the same system type. In addition, defects can result from the interplay of multiple instances of the same system type with instances of other system types. These defects often remain covert as the type-level specification does not explicitly state that there might be one or more other systems of the same type participating. Hence, manual validation must consider these situations on the instance-level. However, it is neither feasible to validate all possible instance-level configurations of collaborating CPS nor possible to investigate all potential situations manually. If one configuration of the collaborative CPS network were left out, this could also lead to defects remaining covert.

To summarize, both—pure type-level and instance-level modeling—have their advantages and can be used in combination. This allows for a complete specification and a closer investigation of certain aspects. However, there is still a need to support validation of model-based specifications with an abstraction that allows investigating a complete specification (i.e., as is done in type-level specifications) and considering the interplay of multiple instances of the same type (i.e., as is done with instance-level diagrams). Therefore, Section 5 proposes the purpose-specific definition of ml-MSCs.

With ml-MSCs it shall be possible to identify functional inadequacies that otherwise remain covert. For instance, a validator might be interested in the effects an intruder vehicle has on the formed platoon. Therefore, it is of interest to investigate all possible platoon configurations to ensure proper platoon behavior in every relevant situation. However, that yields in an inconsiderable number of diagrams to be investigated and will typically also not be necessary as the validator is interested in the direct effects of the intrusion. Hence, it is sufficient to investigate the interplay between platoon leader, the intruder, the vehicle following the intruder and other vehicles in the platoon. While it is necessary to investigate the first three roles in concrete, the latter is used to represent all other vehicles of the platoon and is therefore on an abstract level.

5. MULTI-LEVEL MSC MODELING

To overcome the limitations of only using type-level *or* instance-level during manual validation of system behavior, we propose the systematic specification of multi-level MSCs (ml-MSC). Using a combination of type-level and instance-level information, ml-MSCs can be subjected to inspections or reviews as they show the holistic system specification without overloading the validator with instance-level information. The combination of types and instances within one single model is beneficial as it allows to investigate multiple scenarios and configurations at once. At the same time it allows placing emphasis on important individual aspects of operational situations.

After giving an overview on ml-MSC in Section 5.1, Sections 5.2 and 5.3 will define two distinct processes for creation of ml-MSCs depending on the individual purpose. Either instance-level models can be partly abstracted such that types are introduced within the original instance-level model (see Section 5.2) or type-level models are partially concretized by introducing instance-level elements (see Section 5.3). Section 5.4 summarizes how mixed abstraction MSCs aid validation.

5.1. Overview

To allow systematic generation of ml-MSC two different specification processes are reasonable, as we exclude any unsystematic

ad hoc creation which bears the risk of missing important purpose-specific aspects. The two processes are as follows:

- *Selective Abstraction.* The modeler envisions a concrete scenario that serves the purpose intended to be investigated. System instances and message instances are subsequently investigated to check whether they are needed to represent the concrete defect (or other purpose of investigation) or not. In the latter case, messages and instances are abstracted to a type-level, which allows (a) the focusing on the relevant parts and (b) limits the number of concrete situations that must be investigated. This process is described in Section 5.2.
- *Selective Instantiation.* The modeler starts with a type-level MSC, which can, e.g., be taken from the system specification. Depending on the purpose to be investigated instantiation is used to concretize certain system lifelines and messages as far as needed. This process is described in Section 5.3.

For both approaches it is important to differentiate between the set of type-level elements, i.e., system lifelines and generic messages (for bMSCs, these are typically specified as $b = (I_b, M_b, E_b, \leq_b, \alpha_b)$) and the set of all possible instantiations thereof. This means that I_b defines a certain set of lifelines that can occur at run time and M_b a certain set of messages that can occur during run time, such that $I_b = (i_1, \dots, i_n) \rightarrow \varphi(i_1) = (i_{1_1}, \dots, i_{1_n}), \dots, \varphi(i_n) = (i_{n_1}, \dots, i_{n_n})$ and $M_b = (m_1, \dots, m_n) \rightarrow \varphi(m_1) = (m_{1_1}, \dots, m_{1_n}), \dots, \varphi(m_n) = (m_{n_1}, \dots, m_{n_n})$. Additionally, $E_b, \leq_b, \alpha_b$ also define certain instance-level elements. This way two sets of elements are defined: a set on the type-level and a set on the instance-level.

In our CACC running example, each $i \in I_b$ must be considered a lifeline defined on the type-level. The lifeline "driver" hence represents different persons (e.g., Maria, Frank, Walter, Jane, etc.). These persons are instances of the type "driver" and a common behavior is assumed, as defined by the bMSC. The lifeline "Other CACC" refers to a set of possible real-world instances of a CACC which can act as the lifeline other CACC. However, in this case it cannot be ensured that there will exist only one other CACC our system is interacting with. Hence, the possibility of multiple instances that simultaneously replace other CACCs must be considered. Furthermore, type-level messages are defined. For example, it is defined that the CACC and the brake controller exchange the message "v" (for velocity) such that v defines a set of possible concrete messages that can be exchanged (e.g., messages consisting of a speed value measured in mph between "6" and "60"). In addition, relations between messages and lifelines are also defined on the type-level. For example, after the driver (which means any instance of the type "driver") brakes, the brake controller sends a message "stop maintaining speed" to the CACC in order to end operation. This is a message on type- and instance-level as no other instantiations thereof are expected to exist. In some cases, relations between elements on the type-level depend on instance-level values. For example, v is exchanged and afterwards, it is decided whether another message is sent, depending on v. This might be relative, i.e. "if v < v' send ACCEL else send DECEL" or absolute, i.e. "if v< 20km/h then TERMINATE."

5.2. Selective Abstraction from Instances

To create ml-MSCs as described in the previous section using the process of *selective abstraction*, instance-level models are used as starting point. Subsequently, single instance-level elements are replaced by their respective type. Note, that we do not refer to enriching instance-level models by annotating the type of an instance using the UML "is kind of" relationship, but to purposefully using modeling elements on a type-level and on an instance-level within the same diagrammatic representation. We do so by grouping together selected instances and thus abstracting to the level of system types in order to highlight more generic properties. In many cases, this may include the introduction of roles: Concrete instances are described in terms of the respective role they fulfill, which then corresponds to a type-level description (see [104] for a detailed discussion of the difference between instances, types, and roles, which concludes that roles can be comparably useful on the type-level as well as on the instance-level). Although systems fulfilling a specific role typically instantiate a certain system type, they do not specify the system's instances but abstract groups of system instances that shall behave in a certain situation in the specified way. We can hence replace a concrete instance-level lifeline i_{k_x}, where $i_{k_x} \in (\varphi(i_k) = (i_{k_1}, \dots, i_{k_n}))$, with its corresponding type-level lifeline i_k. The consequence of introducing types on the instance-level is a bMSC, which contains fewer messages and lifelines than the concrete instance-level model. This may help reducing the intra-diagram complexity, while maintaining the same level of detail for the purpose of validation, as the model is easier to comprehend, and potential hazards become more obvious.

Figure 6 shows these benefits using the CACC example. For the CACC example, roles might be "leading vehicle" and "following vehicle." Hence, a type-level specification might distinguish between systems of the different roles. For an instance-level bMSC such as shown in Figure 5, lifelines representing systems fulfilling a specific role are replaced by the role name. The bMSC then describes interactions between systems of the same system type (i.e., the CACC under development) from the perspective of the respective role that is relevant for the concrete instance-level scenario. As can be seen from Figure 6, this leads to the identification of more and less relevant parts for a validation. For instance, it becomes obvious that potential defect occur in the interplay of just four CACCs, the leading vehicle, the following vehicle, and two vehicles driving in parallel and therefore blocking the left lane. Other involved ACCs just exhibit routine behavior. Hence, the bMSC from Figure 6 can consequently be reduced to the bMSC shown in Figure 7, only showing the relevant excerpt from Figure 6. In Figure 7, irrelevant lifelines from Figure 6 are crossed out from the miniature representation thereof to highlight this. Merged and reused lifelines as well as messages are indicated with the dotted transition arrows.

Benefits of this diagrammatic representation lie in the potential to explicitly specify the desired and the undesired behavior of systems involved in a collaboration. This cannot be achieved on the type-level as the collaboration of multiple systems of the same type must be considered. Moreover, this cannot be achieved on the

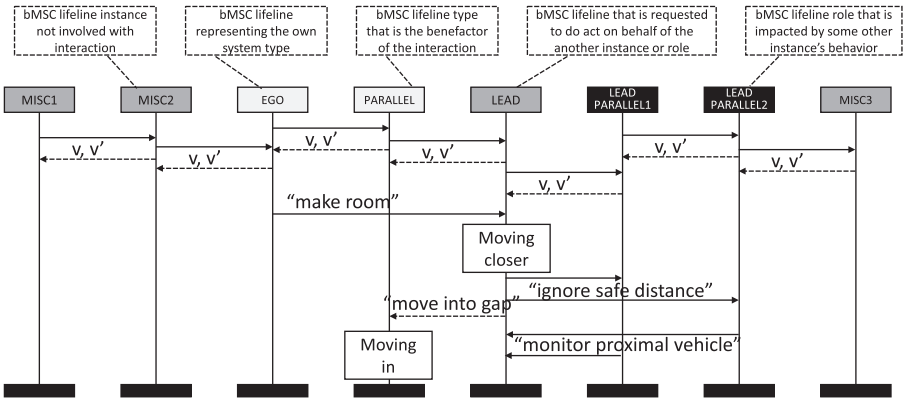

Figure 6 | Multi-level MSC depicting the "Make Room" maneuver. Like before, types abstracted from instances are depicted as light gray, instances as dark gray. Additionally, black lifelines with white labels represent abstracted roles. Again, color shading added for increase clarity.

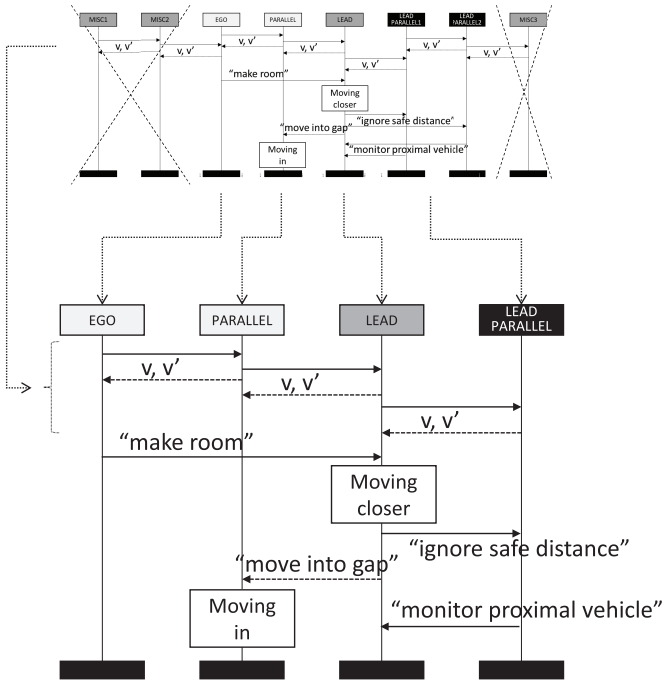

Figure 7 | The roll-reduced multi-level MSC from Figure 6 showing two types (light gray), one instance (dark gray), and one role-reduced type (black). The dotted arrows and dashed crosses show the selective abstraction from irrelevant information depicted by the miniature of Figure 6.

instance-level alone, as the defined behavior shall be subject to all relevant situations and not only to the single one depicted on the instance-level. The defined nominal behavior can then be validated at design time or run time (e.g., [105]).

5.3. Selective Instantiation of Types

ml-MSCs can also be derived by *selective instantiation* of types. In this case, type-level models are the starting points. This way, concrete instances are derived from system types and introduced, where beneficial, to increase the level of detail on run time specific interactions on the instance-level. For example, this may include known and common off-nominal behavior of human agents or systems [105,106].

Like in Section 5.2, the resulting model incorporates elements on the type-level, elements on the instance-level, and potentially the definition or roles. However, in contrast to Section 5.2, the focus is on expanding the level of detail to allow for a more thorough investigation of interactions in which several concrete instances of the same type are involved.

ml-MSCs created in this fashion, hence, mostly contain bMSC lifelines that specify system types. In addition, bMSC lifelines that show system instances are used to emphasis specific aspects in detail. Furthermore, while many messages will be defined on a type-level for reducing complexity and allowing for general statements, some messages will define concrete instances of this message. Hence, in this case we start with a bMSC on the type-level and replace some type-level elements such as the lifeline i_k with "interesting" representatives on the instance-level, i.e., with lifelines of the set $\varphi(i_k) = (i_{k_1}, \ldots, i_{k_n})$. "Interesting" here refers to operational situations that are deemed important by validator and shall be investigated closer. Replacing type-level elements by instance-level elements can thus lead to multiple ml-MSCs to be investigated. For instance, if the validator would like to assess two operational situations for potential defects, two ml-MSCs are created, one for each situation. For another example, the validator may assume that very high or very low values might lead to a critical situation. Therefore, both situations shall be investigated with ml-MSCs. In consequence, the number of ml-MSCs can be higher than the number of corresponding type-level MSCs. However, as still many elements are kept on type-level and only the potentially hazardous influences are instantiated, the number of ml-MSCs to be investigated is still by a large extent smaller then when investigating all possible instance-level combinations.

An example is shown in Figure 8, in which a simplified excerpt of the control algorithm governing the calculation of the vehicle's own speed is shown. While in a type-level model would typically show a single bMSC lifeline "Wheel Sensor" to depict the messages the CACC receives from this kind of sensor, in Figure 8 this lifeline has been instantiated four times to account for all four sensors attached to each wheel of a vehicle. This allows describing a situation that

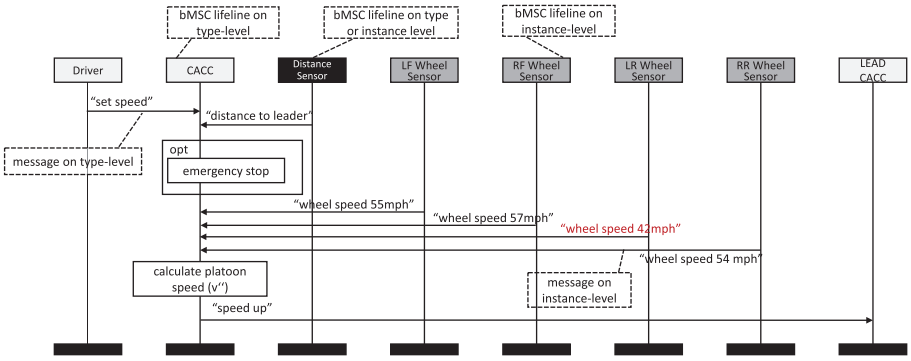

Figure 8 Multi-level MSC showing robust vehicle speed determination. Type-level lifelines are light gray, instance-level are dark gray. Role-type lifelines may be understood as existing on type- and instance-level concurrently, depicted in black.

different sensors produce different signals, which, e.g., shows the need to balance differing values (e.g., through a failure-robust voting algorithm like [107]). Since in type-level models, these four instance-level lifelines would be represented as a single type-level lifeline, the need for value balancing remains covert.

Furthermore, Figure 8 shows type-level and instance-level messages mixed in one diagram. For example, it is specified that the distance sensor sends a message of the type "distance to leader," i.e., no concrete values that are transmitted are prescribed. In contrast, the four wheel sensors transmit concrete values. This supports in the detection of defects resulting from, e.g., sensor failure or runaway values measured by an intact sensor, and helps in common cause analysis and hazard mitigation.

5.4. Summary

ml-MSCs allow the dedicated investigation of system failures that threaten the collaborative operation during run time. ml-MSC, thereby, guide the stakeholders during type-level scenario validation by making use of instance-level information. These instance-level information are used to make the context of some collaborative functionality within the same system (i.e., one CACC) as well as collaborative systems (between several CACCs) explicit. It furthermore opens the possibility to document the effective micro-structure of a collaborative CPS network architecture, i.e., how the collaborative functionality is achieved through the interplay of instances between individual cyber-physical vehicles.

6. EXPERIMENT DESIGN

Throughout this paper, we have argued that the purpose-specific interrelation of instance-level and type-level information into one scenario diagram as suggested in our ml-MSC approach is beneficial to identify functional inadequacies that arise from the interplay of types and instances in automotive CPS. After applications to industrial case examples and discussions with industry partners to ensure the applicability of the proposed solution approach, we designed a controlled experiment to determine whether the use of ml-MSCs showing different collaborative system compositions is beneficial for the manual validation of diagrams depicting system collaboration.

6.1. Experiment Tasks

Participants were asked to conduct a review of three bMSCs (one each of the aforementioned types) in random order. The participants reviewed excerpt diagrams representing a part of the specification of the collision avoidance system in different diagrammatic representations. In each reviewing task, twelve natural language stakeholder intentions were given. For each intention the participant had to decide whether the intention was correctly or incorrectly displayed in the respective bMSC or if it is impossible to tell from the diagram. In addition, the participants were asked to rate their confidence in decision-making on a 5-point Likert scale. After review, each participant completed a questionnaire asking for demographic information and for their attitude toward the perceived support for the review task rendered by the different representations.

6.2. Variables

From the experimental task, it follows that the independent variable (IV) is which diagrammatic representation was used to validate the specified behavior against the stakeholder intentions. This IV has three levels:

- Type-Level MSC (TL): The participants make their decisions based on a type-level MSC as the diagrammatic representation.
- Instance-Level MSC (IL): The participants make their decisions based on an instance-level MSC as the diagrammatic representation.
- ml-MSC (ML): The participants make their decisions based on a ml-MSC as the diagrammatic representation.

To assess the performance of each level of IV, we measured the following dependent variables (DVs):

- Expressiveness: Ratio of stakeholder intentions the participants indicated as suitably depicted in the review artifact and the stakeholder intentions the participants indicated were not suitably depicted the diagrammatic representations.

- Effectiveness: Ratio of correct and incorrect decisions regarding the question whether a stakeholder intention is accurately depicted in the diagrammatic representation.
- Efficiency: Average time needed (in seconds) to make a correct decisions regarding the question whether a stakeholder intention is accurately depicted in the diagrammatic representation.
- User confidence: Average confidence a participant claims for the correctness a decision made.
- Subjective supportiveness: Average result of self-rated standardized questionnaire items from the TAM3 (Technology Acceptance Model v.3, see [108]) for perceived usefulness, perceived ease of use, and computer self-efficacy.

To investigate effects resulting from participants' experience and knowledge, we also measured several covariates, i.e., educational achievements, work experience, and participants' self-rated experience in four categories related to conducting reviews in general and the used modeling notation in particular.

6.3. Hypotheses

For each dependent variable, we derived a null and alternative hypotheses, which are given by Table 1. In case the alternative hypothesis is accepted, we investigate the differences between factors by planned comparison [109]. For this, Table 1 shows subhypotheses for each factor. In the following, we will report the results from the conducted one-way repeated measures ANOVAs and highlight more fine-grained findings regarding the factor-specific alternative hypotheses.

6.4. Participants

The experiment was conducted using domain experts for model-based engineering of embedded systems with varying years of experience. The use of a mixed participant group was chosen to simulate different levels of expertise, as it has been shown that different experiences lead to different results in comparable controlled experiments (cf. [110]). In total, 20 participants took part in the experiment.

6.5. Experiment Material

We used an industrial sample specification from the avionics[3] several features. Domain featuring a collaborative component. Specifically, we chose a collision avoidance system (i.e., a system like [111], which identifies aircraft on a collision course and manipulates flight routes of the aircraft to increase separation altitude). For the purpose of this experiment, bMSCs on the type-level, instance-level, and ml-MSCs were created based on our partners' case example descriptions. Technical jargon as well as intellectual property identifying our industry partners were removed.

[3] A case example from the avionics domain was available due to the nature of the research project in which this work was created. We chose an avionics case example in order to minimize participant bias from automotive systems they may be familiar with, e.g., from their own vehicles. The specification was validated in close collaboration with partners from the automotive and avionics industry to ensure generalizability of results.

6.6. Procedure

The study was conducted as a 30-minute within-subjects online experiment to reduce the negative impact on the participants' perception of anonymity in the experiment. After informed consent was collected and the nature of the study was explained, a trial order (of the three types of bMSCs) was generated for each participant, and the participants were asked to familiarize themselves with the principle functioning of a collision avoidance system as well as the safety-relevant stakeholder intentions pertaining thereto. When participants were ready, they started the review of the bMSC and made their judgments of whether or not some stakeholder intention was correctly or incorrectly depicted in the diagram, or whether the diagram does not allow such a judgment. This means each participant reviewed 36 stakeholder intentions, twelve in TL, twelve in IL, and twelve in ML. For each stakeholder intention, participants were asked to rate their confidence. Once all 36 judgments were complete, participants indicated their preference for each of the three bMSC versions on TAM3 items.

7. EXPERIMENT RESULTS

After data collection concluded, the data were downloaded from the experimental platform. Incomplete and irregular data sets (i.e., a participant that did not finish the experiment or answered in patterns) were discarded. A total of 18 data sets remained. After confirmation of normal distribution, one-way repeated measures ANOVAs were conducted to test if the data shows significant differences between the means of expressiveness (H1), effectiveness (H2), efficiency (H3), confidence (H4), and supportiveness (H5) with regard to the diagram type. In the following, we will discuss answer frequencies and hypothesis test results for each dependent variable and decide whether to reject the corresponding null hypotheses and accept the alternative one.

7.1. Descriptive Statistics

In the following, we report on answer frequencies in order to gain an understanding which representation (TL, IL, or ML) was rated as better, depending on DV. Descriptives are summarized in Table 2. In the following, we discuss the means (μ), standard deviations (σ), and standard error ($\sigma_{\bar{x}}$) for each DV. Not that one participant neglected to respond to questions pertaining to user confidence, which is why descriptive statistic consider one less sample in Table 2.

Expressiveness. Regarding expressiveness, we asked for each stakeholder intention, whether the participant thinks that this question can be answered using the diagrammatic representation shown. *TL* was seen as the least expressive with an average of 68.52% of stakeholder intentions found to be answerable. *IL* was seen as 81.48% and *ML* as 84.72% expressive.

Table 1 | Tested null and alternative hypotheses.

Var	ID	Hypothesis
	H1-0	There is no difference in expressiveness between TL, IL, and ML.
	H1-a	There is a significant difference in expressiveness between TL, IL, and ML.
H1: Expressiveness	H1-a1	There is a significant difference in expressiveness between TL and IL.
	H1-a2	There is a significant difference in expressiveness between TL and ML.
	H1-a3	There is a significant difference in expressiveness between IL and ML.
	H2-0	There is no difference in effectiveness between TL, IL, and ML.
	H2-a	There is a significant difference in effectiveness between TL, IL, and ML.
H2: Effectiveness	H2-a1	There is a significant difference in effectiveness between TL and IL.
	H2-a2	There is a significant difference in effectiveness between TL and ML.
	H2-a3	There is a significant difference in effectiveness between IL and ML.
	H3-0	There is no difference in efficiency between TL, IL, and ML.
	H3-a	There is a significant difference in efficiency between TL, IL, and ML.
H3: Efficiency	H3-a1	There is a IL difference in efficiency between TL and IL.
	H3-a2	There is a significant difference in efficiency between TL and ML.
	H3-a3	There is a significant difference in efficiency between IL and ML.
	H4-0	There is no difference in user confidence between TL, IL, and ML.
	H4-a	There is a significant difference in user confidence between TL, IL, and ML.
H4: User Confidence	H4-a1	There is a significant difference in user confidence between TL and IL.
	H4-a2	There is a significant difference in user confidence between TL and ML.
	H4-a3	There is a significant difference in user confidence between IL and ML.
	H5-0	There is no difference in subjective supportiveness between TL, IL, and ML.
H5: Subjective	H5-a	There is a significant difference in subjective supportiveness between TL, IL, and ML.
Supportiveness	H5-a1	There is a significant difference in subjective supportiveness between TL and IL.
	H5-a2	There is a significant difference in subjective supportiveness between TL and ML.
	H5-a3	There is a significant difference in subjective supportiveness between IL and ML.

Effectiveness. For effectiveness, the ratio of correct answers was measured. *ML* resulted in most correct answers ($\overline{x} = 64.35\%$), followed by *IL* with $\overline{x} = 58.33\%$. Least effective was *TL* with $\overline{x} = 50.46\%$.

Efficiency. Regarding efficiency, we determined the average time used for making one correct decision. Most efficient was *ML* with only 24.51 seconds needed for making a correct decision. *TL* ranked second with $\overline{x} = 31.21sec$, and *IL* ranked last with $\overline{x} = 36.94sec$.

User confidence. The participants were most confident in decision making using *TL* with $\overline{x} = 4.19$ (where 5 means "'very confident'" and 1 means "'least confident'"), followed by *ML* with $\overline{x} = 4.06$ and *IL* with $\overline{x} = 3.99$.

Subjective supportiveness. On average participants found *ML* most supportive with $\overline{x} = 3.45$ (where 5 means "very supportive" and 1 means "not suppportive at all"). *TL* ($\overline{x} = 3.27$) and *IL* ($\overline{x} = 3.13$) were seen less supportive. As subjective supportiveness is calculated from the average of multiple items, reliability of the measurement must be ensured, although we used the TAM3. Therefore, we calculated Cronbach's alpha showing optimal reliability of the measurements. For *ML* $\alpha(10) = .962$, for *TL* $\alpha(10) = .951$, for *IL* $\alpha(10) = .938$.

7.2. Hypotheses Tests

In Section 7.1, we have shown how the different representation types perform with regard to the respective DV. However, in order to determine which of these differences is significant, we computed a one-way repeated measures ANOVA to test our hypothesis and followed up significant results with post hoc tests. Table 3 provides an overview of the results from the one-way repeated measures ANOVA.

Expressiveness. A one-way repeated measures ANOVA showed a significant difference in the expressiveness of *TL*, *IL*, and *ML*. $F(2, 38) = 14.83$, $p < .001$. We therefore reject H1-0 and accept H1-a.

Post hoc tests using the Bonferroni correction revealed that expressiveness of *ML* ($M = 85.83\%$) is significantly higher than the expressiveness of *TL* ($M = 67.5\%$), $p < .001$. Furthermore, the expressiveness of *IL* ($M = 80.83\%$)is significantly higher than the expressiveness of *TL* $p = .003$. We can therefore accept H1-a2 and H1-a3

Effectiveness. The results from a one-way repeated measures ANOVA showed a significant difference in the effectiveness of

Table 2 Descriptive statistics.

		N	Mean	Std. Deviation	Minimum	Maximum
Expressiveness	TL	18	68.52%	12.96%	50.00%	100.00%
	IL	18	81.48%	13.27%	50.00%	100.00%
	ML	18	84.72%	12.54%	58.33%	100.00%
Effectiveness	TL	18	50.46%	8.80%	33.33%	58.33%
	IL	18	58.33%	15.12%	25.00%	83.33%
	ML	18	64.35%	17.57%	33.33%	91.67%
Efficiency	TL	18	31.21	16.81	11.57	72.29
	IL	18	36.94	23.82	12.14	93.57
	ML	18	24.51	17.16	7.00	63.25
User confidence	TL	18	4.19	0.83	2.00	5.00
	IL	17	3.99	0.90	1.00	5.00
	ML	18	4.06	0.99	1.00	5.00
Subjective supportiveness	TL	18	3.27	0.84	1.10	4.30
	IL	18	3.13	0.71	1.90	4.40
	ML	18	3.45	0.89	1.20	5.00

Table 3 Results from one-way repeated measures ANOVA.

	Sum of Squares	*df*	Mean Square	*F*	Sig.
Expressiveness	3592.593	2	1796.296	14.833	0.000
Effectiveness	2918.981	2	1459.491	9.011	0.001
Efficiency	1681.486	2	840.743	1.551	0.225
Confidence	0.336	2	0.168	0.800	0.457
Subjective supportiveness	2.344	2	1.172	2.039	0.144

TL, *IL*, and *ML* $F(2, 38) = 9.01$, $p < .001$. We therefore reject H2-0 and accept H2-a.

Post hoc tests indicated that the effectiveness of *ML* ($M = 67.09\%$) is significantly higher than the effectiveness of *TL* ($M = 50.00\%$), $p = .002$. Furthermore, the effectiveness of *IL*($M = 58.75\%$) is significantly higher than the effectiveness of *TL* $p = .033$. We can therefore accept H2-a1 and H2-a2.

Efficiency. There were no significant differences in the efficiency for *TL*, *IL*, and *ML* $F(2, 38) = 1.55$, $p = .23$. Therefore, we cannot reject H3-0.

User confidence. The test showed no significant difference in the user confidence for *TL*, *IL*, and *ML* $F(2, 36) = 0.80$, $p = .46$. We therefore cannot reject H4-0.

Subjective supportiveness. The results of a one-way repeated measures ANOVA show that participants did not rate the diagrammatic representations' subjective supportiveness significantly different $F(2, 38) = 2.04$, $p > .14$. We therefore cannot reject H5-0.

8. DISCUSSION

In the following, we discuss what the experimental results from Section 7 mean with regard to information being presented in type-level, instance-level, or multi-level MSCs for the validation of collaborative CPS.

8.1. Major Findings

Table 4 provides an overview of the accepted hypotheses and the findings where we were unable to establish a significant difference between representation types. In the following, we will discuss findings regarding TIL, findings regarding the difference between Type and Instance, and interpretations based on both findings.

Benefits of using ml-MSCs. To sum up results regarding *ML*, we can state that:

- *ML* is highly significantly more expressive than *TL*.
- *ML* is highly significantly more effective than *TL*.
- *ML* is also slightly more expressive than *IL* (although the experiment could not detect a statistically significant difference).
- *ML* is also more effective than *IL* (although the experiment could not detect a statistically significant difference).
- *ML* is most efficient and seen as most supportive (although the experiment could not detect a statistically significant difference).

These findings show, that *multi-level MSCs* are advantageous compared to bMSCs only containing *type-level* or *instance-level* information. *ml-MSCs* are more expressive and effective when used for

Table 4 Results of hypotheses tests.

Hypothesis		Description	Result
H1 (expressiveness)	H1-0	There is no difference in expressiveness between TL, IL, and ML.	Accepted
	H1-a	There is a significant difference in expressiveness between TL, IL, and ML.	
	H1-a2	*ML is highly significantly more expressive than TL*	
	H1-a3	*IL is highly significantly more expressive than TL*	
H2 (effectiveness)	H2-0	There is no difference in effectiveness between TL, IL, and ML.	Accepted
	H2-a	There is a significant difference in effectiveness between TL, IL, and ML.	
	H2-a1	*IL is significantly more effective than TL*	
	H2-a2	*ML is highly significantly more effective than TL*	
H3 (efficiency)	H3-0	There is no difference in efficiency between TL, IL, and ML.	Cannot be rejected
	H3-a	There is a significant difference in efficiency between TL, IL, and ML.	
H4 (user confidence)	H4-0	There is no difference in user confidence between TL, IL, and ML.	Cannot be rejected
	H4-a	There is a significant difference in user confidence between TL, IL, and ML.	
H5 (subjective supportiveness)	H5-0	There is no difference in subjective supportiveness between TL, IL, and ML.	Cannot be rejected
	H5-a	There is a significant difference in subjective supportiveness between TL, IL, and ML.	

reviews. In addition, *ml-MSCs* is also more expressive and effective compared to *instance-level* bMSCs. Although this is not by a significant margin, we assume that this effect is indeed present but with a smaller effect size compared to *type-level* bMSCs. Likely, due to the limited number of participants, significance was not reached.

Furthermore, *ml-MSCs* is most efficient and most supportive compared with *type-level* and *instance-level* bMSCs. Hence, we can conclude that the use of *multi-levle MSCs* supports manual validation better than the use of *type-level* or *instance-level* bMSCs.

Relationship between type-level and instance-level bMSCs. To sum up results regarding *type-level* and *instance-level* bMSCs, we found that:

- *IL* is highly significantly more expressive than *TL*.
- *IL* is significantly more effective than *TL*.
- *TL* leads to most confident results (although the experiment could not detect a statistically significant difference).
- *TL* is more efficient and supportive than *IL* (although the experiment could not detect a statistically significant difference).

These findings show, while *instance-level* bMSCs are more expressive and effective than *type-level* bMSCs, using diagrams on the *type-level* seems to be more efficient, leads to more confident decisions, and is seen as more supportive. Hence, it can be concluded that *instance-level* bMSCs do have other, subjective advantages over *type-level* bMSCs and vice versa (which were beyond the scope of this investigation).

Interpretation. In comparison of type-level and instance-level diagrams, we see that bMSC on the instance-level are more expressive and more effective but bMSC on a type-level are more efficient, users' are more confident when using type-level MSC and find it more supportive for their tasks. These findings concur with findings from previous work comparing type-level and instance-level MSCs of different size [64].

This experiment has shown that multi-level MSCs combine the benefits of MSC on instance-level (i.e., they are expressive and efficient) and type-level (i.e., they are efficient, user confidence increasing, and supportive). Furthermore, they are—if the used abstractions are chosen appropriately—even more expressive and effective than MSC on an instance-level and more efficient than MSC on a type-level. However, there seems to remain a minor disadvantage compared to MSC on type-level when it comes to user confidence and subjective supportiveness. It is to note that these differences were not statistically significant.

8.2. Threats to Validity

Despite our efforts to carefully design our empirical validation of ml-MSCs, as with any empirical study, some threats to validity remain. These are discussed in the following.

Internal validity. The mode of experimentation may have allowed some participants to not take the experiment seriously and answer in patterns. To address this, we checked all data sets carefully and discarded incomplete and irregular responses. Thus we are confident that our conclusions are only based on adequate responses. Another threat, however, is possible researcher bias in selection of the selected stakeholder intentions. To address this issue, we asked academic and industry collaborators to validate our selection. We are reasonably confident that this minimized bias, but may possibly not have entirely eliminated it.

External validity. To foster external validity, we recruited participants from typical conceptual modeling backgrounds. To what degree these results are generalizable to members of the industry with less modeling background is an open question, however we feel confident that the modeling task reported herein is industry-typical, as confirmed by our industry partners. Moreover, we used an industry-realistic case example (albeit not a real one) to from the avionics domain. While we took great care to ensure generalizability to the automotive domain (and, conceivably, other industries), we do not yet have evidence to what degree these results generalize.

Conclusion validity. Conclusion validity is threatened by inherent researcher bias, as we had an interest in showing the usefulness of multi-level MSC. We hence adopted a strict threshold of 95% and reported positive and negative results. Consequently, as has been shown in the experiment results, descriptive statistics in many cases indicate differences, which turned out to be not statistically significant, which we assume is a result of the limited number of participants. While we strive for a repetition with more participants this is a challenging task taking availability of experienced industrial volunteers into account.

Construct validity. We used an experimental setup, which has also been used in many other published studies (see Section 2.3). In addition, experimental materials were discussed with industry partners to ensure not only generalizability but also adequacy of the limited representation format of an online questionnaire. The use of online questionnaire poses another threat to construct validity, which we opted for due to the benefits for participant recruitment.

8.3. Deductions and Inferences

Under consideration of the threats to validity and the fact that some statistical tests did not yield a level of significance but that findings well align with previous experiments, we are confident to claim that the purposeful use of multi-level MSC is beneficial compared to the use of only either type-level MSC or instance-level MSC. As we have already shown that MSC in general are good language to support manual validation tasks [57,59], this indicates that the use of multi-level MSC can considerably support validation.

This seems to be particularly true for collaborative automotive CPS, where the sheer number of instances to be considered during validation of functional behavior becomes infeasible to investigate manually. Therefore, ml-MSC can be used to limit the size of the validation task while at the same time increasing effectiveness and efficiency of the investigation.

However, it remains unclear why increased expressiveness, effectiveness, and efficiency did not yield in significantly increased user confidence and subjective supportiveness. As this is a threat to manual validation tasks—since the perception of the reviewer does not fit the actual performance—future work remains to investigate this threat and to propose alleviation. Nevertheless, we found the proposed use of ml-MSC most beneficial when it comes to manual validation in the first place.

9. CONCLUSION

In this paper, we have argued that for validation of the functional behavior of collaborative automotive CPSs, the inspection of type-level models is insufficient as some defects do only occur in specific interaction scenarios. Therefore, the validator needs to investigate the instance-level to adequately consider and detect these potential functional inadequacies. However, due to the variety of different compositions to be considered in instance-level scenarios, an amount of instance-level models would need to be investigated that is not feasible to handle. To this end, we have investigated whether the use of multi-level models can mitigate this issue.

We contributed the theoretical underpinnings and two processes to systematically derive ml-MSCs. ml-MSCs are a variation of traditional ITU MSCs that allows for the specific interrelation of instance-level and type-level information within the same diagram. We have proposed that their use is particularly beneficial for manual validation of collaborative automotive CPS functionality, as their development makes it necessary to investigate the anticipated behavior of different types of systems and also the interactions between several system instances of different types at run time.

To investigate whether the use of multi-level modeling can aid in validation, we conducted a controlled experiment using the proposed ml-MSCs. We compared the use of ml-MSCs to MSCs on type- and instance-level regarding expressiveness, effectiveness, efficiency, confidence, and subjective supportiveness. Results showed that using ml-MSCs during reviews of collaborative CPS specifications allows for significantly easier and more accurate identification of defects. However, there was no impact on time needed for inspection, or the users' confidence or the subjective supportiveness compared to instance-level or type-level representations. Thus, we can conclude that ml-MSCs lead to significantly more expressive review artifacts and significantly more effective reviews compared to type-level MSCs, while at the same time the amount of ml-MSCs is considerably smaller than when using instance-level MSCs. Hence, we found ml-MSCs to be a good means to support the validation of collaborative CPS.

While we envision ml-MSCs to be particularly useful in collaborative runtime settings as outlined in Section 1, in principle, the fundamentals presented in this manuscript can be applied in more general, non-CPS settings as well. Thus, future work may be concerned with exploring the impact of ml-MSCs and multi-level modeling in more detail, particularly with more rigorous experimentation using higher sample size and industry representatives. We hope that this will help us identify good practices for the creation of ml-MSCs and the definition characteristics to define what makes an ml-MSC a good review artifact for inspections. Moreover, extending ml-MSCs to consider timing requirements, e.g., by extending Timed Automata [82] with the concepts presented herein may be a promising avenue to semi-automatically uncover certain types of defects (e.g., convoy speed propagation delay in large CPS networks [23]).

CONFLICTS OF INTEREST

The authors declare that they have no conflict of interest. All authors contributed equally.

Funding Statement

This research was partly funded by the German Federal Ministry of Education and Research (grant no. 01IS12005C and grant no. 01IS16043V).

ACKNOWLEDGMENTS

We like to thank our industrial partners for their support with case examples and in creation of the experiment material used. Particularly, we thank Peter Heidl, Jens Höfflinger and John MacGregor

(Bosch), Frank Houdek (Daimler), Stefan Beck and Arnaud Boyer (Airbus), as well as Thorsten Weyer (Schäffler).

REFERENCES

[1] Lutin, J. Not If, but When: autonomous driving and the future of transit, J Public Transp 2018;21;92–103. https://scholarcommons.usf.edu/jpt/vol21/iss1/10.

[2] Berger, C, Rumpe, B. Autonomous driving - 5 years after the urban challenge: the anticipatory vehicle as a cyber-physical system. In: Goltz, U, Magnor, M, Appelrath, HJ, Matthies, HK, Balke, WT, Wolf, L, editors. INFORMATIK 2012, Bonn, Germany: Gesellschaft für Informatik e.V.; 2012, p. 789–98.

[3] Mao, J, Chen, L. Runtime monitoring for cyber-physical systems: a case study of cooperative adaptive cruise control. In 2012 Second International Conference on Intelligent System Design and Engineering Application, Sanya, China; 2012, p. 509–15.

[4] Ciccozzi, F, Crnkovic, I, Ruscio, DD, Malavolta, I, Pelliccione, P, Spalazzese, R. Model-driven engineering for mission-critical IoT systems. IEEE Softw 2017;34;46–53.

[5] Daun, M, Tenbergen, B, Brings, J, Weyer, T. Documenting assumptions about the operational context of long-living collaborative embedded systems. In Proceedings of 2nd Collaborative Workshop on Evolution and Maintenance of Long-Living Software Systems (EMLS), Gesellshaft fuer Informatik e.V.; Kiel, Germany, 2015, p. 115–7.

[6] Ishigooka, T, Saissi, H, Piper, T, Winter, S, Suri, N. Practical use of formal verification for safety critical cyber-physical systems: a case study. In 2014 IEEE International Conference on Cyber-Physical Systems, Networks, and Applications, Hong Kong, China: IEEE; 2014, p. 7–12.

[7] Chen, H. Theoretical foundations for cyber-physical systems: a literature review, J Ind Integr Manag 2017;2;1750013.

[8] Shanaa, W, Spier, S, Tenbergen, B. A case study into the development process of cyber physical systems. In Joint Proceedings of REFSQ-2017 Workshops, Doctoral Symposium, Research Method Track, and Poster Track co-located with the 23rd International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2017), Essen, Germany, 2017.

[9] Zheng, X, Julien, C, Kim, M, Khurshid, S. Perceptions on the state of the art in verification and validation in cyber-physical systems. IEEE Syst J 2017;11;2614–27.

[10] Glinz, M. Improving the quality of requirements with scenarios. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.27.6387&rep=rep1&type=pdf.

[11] Glinz, M, Fricker, SA. On shared understanding in software engineering: an essay. Comput Sci Res Dev 2015;30;363–76.

[12] SAE international standard 4761: guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment. 1996.

[13] IEC international standard 61508: functional safety standards. 2010.

[14] ISO international standard 26262: road vehicles - functional safety, Technical Report, 2011. https://www.iso.org/standard/43464.html.

[15] Fagan, ME. Design and code inspections to reduce errors in program development. IBM Syst J 1976;15;182–211.

[16] Fagan, ME. Advances in software inspections. IEEE Trans Softw Eng 1986;12;744–51.

[17] Boehm, BW. Industrial software metrics top 10 list, IEEE Softw 1987;4;84–5. https://ci.nii.ac.jp/naid/10019649808/en/.

[18] Martin, J, Tsai, WT. N-fold inspection: a requirements analysis technique. Commun ACM 1990;33;225–32.

[19] Thelin, T, Runeson, P, Wohlin, C. An experimental comparison of usage-based and checklist-based reading. IEEE Trans Softw Eng 2003;29;687–704.

[20] Flynn, DJ, Warhurst, R. An empirical study of the validation process within requirements determination, Inf Syst J 1994;4;185–212.

[21] Hauer, F, Schmidt, T, Holzmüller, B, Pretschner, A. Did we test all scenarios for automated and autonomous driving systems? In 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand; 2019, p. 2950–5.

[22] Trapp, M, Schneider, D, Liggesmeyer, P. A safety roadmap to cyber-physical systems. In: Münch, J, Schmid, K, editors. Perspectives on the future of software engineering, Berlin, Heidelberg, Germany: Springer; 2013, p. 81–94.

[23] Daun, M, Salmon, A, Tenbergen, B, Weyer, T. Today's challenges and potential solutions for the engineering of collaborative embedded systems. 2015. https://www.semanticscholar.org/paper/Today%E2%80%99s-Challenges-and-Potential-Solutions-for-the-Daun-Salmon/6a882d58bb9b3a72abdef747d3e11d1b0c6c74cb.

[24] Tenbergen, B, Weyer, T, Pohl, K. Hazard Relation Diagrams: a diagrammatic representation to increase validation objectivity of requirements-based hazard mitigations, Requir Eng 2018;23;291–329.

[25] Recommendation ITU-T Z.120: Message Sequence Chart (MSC). Series Z: languages and general software aspects for telecommunication systems. International Telecommunication Union; 2016.

[26] Maoz, S. Polymorphic scenario-based specification models: semantics and applications. Softw Syst Model 2012;11;327–45.

[27] Jahn, M, Roth, B, Jablonski, S. Instance specialization - a pattern for multi-level meta modelling. CEUR Workshop Proc 2014;1286;23–32.

[28] Theisz, Z, Mezei, G. An algebraic instantiation technique illustrated by multilevel design patterns. In Proceedings of the 2nd International Workshop on Multi-Level Modelling co-located with ACM/IEEE 18th International Conference on Model Driven Engineering Languages & Systems (MoDELS 2015), Ottawa, Canada; 2015, vol. 1505, p. 53–62. http://ceur-ws.org/Vol-1505/p6.pdf.

[29] Atkinson, C, Gerbig, R, Kühne, T. A unifying approach to connections for multi-level modeling. In 2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS), Ottawa, Canada; 2015, p. 216–25.

[30] Nguyen, VH, Fouquet, F, Plouzeau, N, Barais, O. A process for continuous validation of self-adapting component based systems. In Proceedings of the 7th Workshop on Models@run.time (MRT'12), Innsbruck, Austria: Association for Computing Machinery; 2012, p. 32–7.

[31] Jordan, A, Mayer, W, Stumptner, M. Multilevel modelling for interoperability. CEUR Workshop Proc 2014;1286;93–102.

[32] Frank, U. Toward a unified conception of multi-level modelling: advanced requirements. In Proceedings of MODELS 2018 Workshops: ModComp, MRT, OCL, FlexMDE, EXE, COMMitMDE, MDETools, GEMOC, MORSE, MDE4IoT,

MDEbug, MoDeVVa, ME, MULTI, HuFaMo, AMMoRe, PAINS Co-located with ACM/IEEE 21st International Conference on Model Driven Engineering Languages and Systems (MODELS 2018), Copenhagen, Denmark; 2018, p. 718–27. http://ceur-ws.org/Vol-2245/multi_paper_10.pdf.

[33] Clark, T, Gonzalez-Perez, C, Henderson-Sellers, B. A foundation for multi-level modelling. In MULTI 2014: Multi-Level Modelling: Proceedings of the Workshop on Multi-Level Modelling co-located with ACM/IEEE 17th International Conference on Model Driven Engineering Languages & Systems (MoDELS 2014), Valencia, Spain: CEUR; 2014, p. 43–52.

[34] Reinhartz-Berger, I, Sturm, A, Clark, T. Exploring multi-level modeling relations using variability mechanisms. In Proceedings of the 2nd International Workshop on Multi-Level Modelling co-located with ACM/IEEE 18th International Conference on Model Driven Engineering Languages & Systems (MoDELS 2015), Ottawa, Canada; 2015, vol. 1505, p. 23–32. http://ceur-ws.org/Vol-1505/p3.pdf.

[35] Partridge, C, Cesare, SD, Mitchell, A, Gailly, F, Khan, M. Developing an ontological sandbox: investigating multi-level modelling\'s possible metaphysical structures.In Proceedings of MODELS 2017 Satellite Event: Workshops (ModComp, ME, EXE, COMMitMDE, MRT, MULTI, GEMOC, MoDeVVa, MDETools, FlexMDE, MDEbug), Posters, Doctoral Symposium, Educator Symposium, ACM Student Research Competition, and Tools and Demonstrations co-located with ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS 2017), Austin, TX, USA; 2017, vol. 2019, p. 226–34. http://ceur-ws.org/Vol-2019/multi_3.pdf.

[36] Cicchetti, A. Proceedings of the 2nd International Workshop on Multi-Level Modelling co-located with ACM/IEEE 18th International Conference on Model Driven Engineering Languages & Systems (MoDELS 2015), Ottawa; Canada, 2015, 1463;12–17.

[37] Gogolla, M. Experimenting with multi-level models in a two-level modeling tool. In Proceedings of the 2nd International Workshop on Multi-Level Modelling co-located with ACM/IEEE 18th International Conference on Model Driven Engineering Languages & Systems (MoDELS 2015), Ottawa, Canada; 2015, vol. 1505, p. 3–12. http://ceur-ws.org/Vol-1505/p1.pdf.

[38] Igamberdiev, M, Grossmann, G, Stumptner, M. A feature-based categorization of multi-level modeling approaches and tools. In Proceedings of the 3rd International Workshop on Multi-Level Modelling co-located with ACM/IEEE 19th International Conference on Model Driven Engineering Languages & Systems (MoDELS 2016), Saint-Malo, France; 2016, vol. 1722, p. 45–55. http://ceur-ws.org/Vol-1722/p4.pdf.

[39] Al-Hilank, S, Jung, M, Kips, D, Husemann, D, Philippsen, M. Using multilevel-modeling techniques for managing mapping information. In Proceedings of the Workshop on Multi-Level Modelling co-located with ACM/IEEE 17th International Conference on Model Driven Engineering Languages & Systems (MoDELS 2014), Valencia, Spain; 2014, vol. 1286, p. 103–112. http://ceur-ws.org/Vol-1286/p11.pdf.

[40] Atkinson, C, Gerbig, R, Tunjic, CV. Enhancing classic transformation languages to support multi-level modeling, Softw Syst Model 2015;14;645–66.

[41] Demuth, A, Riedl-Ehrenleitner, M, Egyed, A. Towards flexible, incremental, and paradigm-agnostic consistency checking in multi-level modeling environments, CEUR Workshop Proc 2014;1286;73–82.

[42] Carvalho, VA, Almeida, JPA. Toward a well-founded theory for multi-level conceptual modeling. Softw Syst Model 2018;17;205–31.

[43] Guerra, E, Lara, J. Towards automating the analysis of integrity constraints in multi-level models. CEUR Workshop Proc 2014;1286;63–72.

[44] Atkinson, C, Kühne, T. On evaluating multi-level modeling. In Proceedings of MODELS 2017 Satellite Event: Workshops (ModComp, ME, EXE, COMMitMDE, MRT, MULTI, GEMOC, MoDeVVa, MDETools, FlexMDE, MDEbug), Posters, Doctoral Symposium, Educator Symposium, ACM Student Research Competition, and Tools and Demonstrations co-located with ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS 2017), Austin, TX, USA; p. 274–77. http://ceur-ws.org/Vol-2019/multi_10.pdf.

[45] Macías, F, Rutle, A, Stolz, V. A tool for the convergence of multilevel modelling approaches. In Proceedings of MODELS 2018 Workshops: ModComp, MRT, OCL, FlexMDE, EXE, COMMitMDE, MDETools, GEMOC, MORSE, MDE4IoT, MDEbug, MoDeVVa, ME, MULTI, HuFaMo, AMMoRe, PAINS co-located with ACM/IEEE 21st International Conference on Model Driven Engineering Languages and Systems (MODELS 2018), Copenhagen, Denmark; 2018, vol. 2245, p. 633–42. http://ceur-ws.org/Vol-2245/multi_paper_1.pdf.

[46] Alam, O, Corley, J, Masson, C, Syriani, E. Challenges for reuse in collaborative modeling environments. In MODELS Workshops. Copenhagen, Denmark, 2018.

[47] Jolak, R, Liebel, G. Position paper: knowledge sharing and distances in collaborative modeling. In 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), Munich, Germany; 2019, p. 415–16.

[48] Dávid, I. A multi-paradigm modeling foundation for collaborative multi-view model/system development. In SRC@MoDELS. 2016. https://research.vu.nl/en/publications/a-multi-paradigm-modeling-foundation-for-collaborative-multi-view.

[49] Nie, K, Yue, T, Ali, S, Zhang, L, Fan, Z. Constraints: the core of supporting automated product configuration of cyber-physical systems. In: Moreira, A, Schätz, B, Gray, J, Vallecillo, A, Clarke, P, editors. Model-driven engineering languages and systems, Lecture notes in computer science, Berlin, Heidelberg, Germany: Springer; 2013, p. 370–87.

[50] Seiger, R, Huber, S, Heisig, P, Aßmann, U. Toward a framework for self-adaptive workflows in cyber-physical systems. Softw Syst Model 2019;18;1117–34.

[51] Vangheluwe, H. Multi-paradigm modelling of cyber-physical systems. In 2018 IEEE/ACM 4th International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS), Gothenburg, Sweden; 2018, p. 1.

[52] Ruchkin, I. Architectural and analytic integration of cyber-physical system models. In Proceedings of the ACM Student Research Competition at MODELS 2015 Co-located with the ACM/IEEE 18th International Conference MODELS 2015, Ottawa, Canada; 2015, vol. 1503, p. 35–40. http://ceur-ws.org/Vol-1503/07_pap_ruchkin.pdf.

[53] Ruchkin, I. Towards integration of modeling methods for cyber-physical systems. In Proceedings of the Doctoral Symposium at the 18th ACM/IEEE International Conference of

Model-Driven Engineering Languages and Systems 2015 (MoDELS 2015), Ottawa, Canada; 2015, vol. 1531. http://ceur-ws.org/Vol-1531/paper9.pdf.

[54] Simko, G, Lindecker, D, Levendovszky, T, Neema, S, Sztipanovits, J. Specification of cyber-physical components with formal semantics - integration and composition. In: Moreira, A, Schätz, B, Gray, J, Vallecillo, A, Clarke, P, editors. Model-driven engineering languages and systems, Lecture notes in computer science, Berlin, Heidelberg, Germany: Springer; 2013, p. 471–87.

[55] Mosterman, PJ, Zander, J. Cyber-physical systems challenges: a needs analysis for collaborating embedded software systems. Softw Syst Model 2016;15;5–16.

[56] Vasilevskaya, M, Nadjm-Tehrani, S, Gunawan, LA, Herrmann, P. Security asset elicitation for collaborative models. In Proceedings of the Workshop on Model-Driven Security (MDsec '12), Innsbruck, Austria: Association for Computing Machinery; 2012.

[57] Daun, M, Weyer, T, Pohl, K. Improving manual reviews in function-centered engineering of embedded systems using a dedicated review model. Softw Syst Model 2019;18;3421–59.

[58] Daun, M, Brings, J, Krajinski, L, Weyer, T. On the benefits of using dedicated models in validation processes for behavioral specifications. In Proceedings of the International Conference on Software and System Processes (ICSSP 2019), Montreal, Canada: IEEE / ACM; 2019, p. 44–53.

[59] Daun, M, Brings, J, Weyer, T. On the impact of the model-based representation of inconsistencies to manual reviews - results from a controlled experiment. In Conceptual Modeling - 36th International Conference (ER 2017), Valencia, Spain; 2017, p. 466–73.

[60] Tenbergen, B, Weyer, T, Pohl, K. Supporting the validation of adequacy in requirements-based hazard mitigations. In: Fricker, SA, Schneider, K, editors. Requirements engineering: foundation for software quality, Lecture notes in computer science, Cham, Switzerland: Springer; 2015, p. 17–32.

[61] Tenbergen, B, Weyer, T. Generation of hazard relation diagrams: formalization and tool support, Softw Syst Model 2021;20; 175–210.

[62] Brandstetter, V, Froese, A, Tenbergen, B, Vogelsang, A, Wehrstedt, JC, Weyer, T. Early validation of automation plant control software using simulation based on assumption modeling and validation use cases. Complex Syst Inform Model Q 2015;4; 50–65.

[63] Daun, M, Salmon, A, Weyer, T, Pohl, K. The impact of students' skills and experiences on empirical results: a controlled experiment with undergraduate and graduate students. In Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering (EASE 2015), Nanjing, China: ACM; 2015, p. 29.

[64] Daun, M, Brings, J, Weyer, T. Do instance-level review diagrams support validation processes of cyber-physical system specifications: results from a controlled experiment. In Proceedings of the International Conference on Software and System Processes (ICSSP 2019), Seoul, Republic of Korea: ACM; 2020, p. 10.

[65] Miller, J, Wood, M, Roper, M. Further experiences with scenarios and checklists. Empir Softw Eng 1998;3;37–64.

[66] Basili, VR, Green, S, Laitenberger, O, Lanubile, F, Shull, F, Sørumgård, S, Zelkowitz, MV. The empirical investigation of perspective-based reading. Empir Softw Eng 1996;1;133–64.

[67] He, L, Carver, JC. PBR vs. checklist: a replication in the n-fold inspection context. In International Symposium on Empirical Software Engineering (ISESE 2006), Rio de Janeiro, Brazil: ACM; 2006, p. 95–104.

[68] Maldonado, JC, Carver, J, Shull, F, Fabbri, SCPF, Dória, E, Martimiano, LAF, Mendonça, MG, Basili, VR. Perspective-based reading: a replicated experiment focused on individual reviewer effectiveness. Empir Softw Eng 2006;11;119–42.

[69] Porter, AA, Votta, LG, Basili, VR. Comparing detection methods for software requirements inspections: a replicated experiment. IEEE Trans Softw Eng 1995;21;563–75.

[70] Porter, AA, Votta, LG. Comparing detection methods for software requirements inspections: a replication using professional subjects. Empir Softw Eng 1998;3;355–79.

[71] Laitenberger, O, Emam, KE, Harbich, TG. An internally replicated quasi-experimental comparison of checklist and perspective-based reading of code documents. IEEE Trans Softw Eng 2001;27;387–421.

[72] Berling, T, Runeson, P. Evaluation of a perspective based review method applied in an industrial setting, IEE Proc Softw 2003;150;177–84.

[73] Sabaliauskaite, G, Kusumoto, S, Inoue, K. Assessing defect detection performance of interacting teams in object-oriented design inspection. Inf Softw Technol 2004;46;875–86.

[74] Luckcuck, M, Farrell, M, Dennis, LA, Dixon, C, Fisher, M. Formal specification and verification of autonomous robotic systems: a survey. ACM Comput Surv 2019;52;1–41

[75] Damm, W, Harel, D. LSCs: breathing life into message sequence charts, Formal Methods Syst Design 2001;19;45–80.

[76] Bohn, J, Damm, W, Wittke, H, Klose, J, Moik, A. Modeling and validating train system applications using statemate and live sequence charts. In Proceedings of Integrated Design and Process Technology, Society for Design and Process Science; Pasadena, California (USA), 2002, p. 1–9.

[77] Klose, J, Wittke, H. An automata based interpretation of live sequence charts. In: Margaria, T, Yi, W, editors. Tools and algorithms for the construction and analysis of systems, Lecture notes in computer science, Berlin, Heidelberg, Germany: Springer; 2001, p. 512–27.

[78] Bontemps, Y, Heymans, P, Schobbens, P. From live sequence charts to state machines and back: a guided tour. IEEE Trans Softw Eng 2005;31;999–1014.

[79] Harel, D, Marelly, R. Playing with time: on the specification and execution of time-enriched lscs. In Proceedings. 10th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems, Fort Worth, TX, USA; 2002, p. 193–202.

[80] Brill, M, Damm, W, Klose, J, Westphal, B, Wittke, H. Live sequence charts. In: Ehrig, H, Damm, W, Desel, J, Große-Rhode, M, Reif, W, Schnieder, E, Westkämper, E, editors. Integration of software specification techniques for applications in engineering: priority program softSpez of the German Research Foundation (DFG), Final report, Lecture notes in computer science, Berlin, Heidelberg, Germany, Springer, 2004, p. 374–99.

[81] Rodríguez, A, Rutle, A, Durán, F, Kristensen, LM, Macías, F. Multilevel modelling of coloured petri nets. In Proceedings of MODELS 2018 Workshops: ModComp, MRT, OCL, FlexMDE, EXE, COMMitMDE, MDETools, GEMOC, MORSE, MDE4IoT,

MDEbug, MoDeVVa, ME, MULTI, HuFaMo, AMMoRe, PAINS co-located with ACM/IEEE 21st International Conference on Model Driven Engineering Languages and Systems (MODELS 2018), Copenhagen, Denmark; 2018, vol. 2245, p. 663–72. http://ceur-ws.org/Vol-2245/multi_paper_4.pdf.

[82] Alur, R, Dill, DL. A theory of timed automata. Theor Comput Sci 1994;126;183–235.

[83] Hilscher, M, Schwammberger, M. An abstract model for proving safety of autonomous urban traffic. In: Sampaio, A, Wang, F, editors. Theoretical Aspects of Computing (ICTAC 2016), Cham, Switzerland: Springer; 2016, p. 274–92.

[84] Schwammberger, M. Introducing liveness into multi-lane spatial logic lane change controllers using uppaal, Electron Proc Theor Comput Sci 2018;269;17–31.

[85] Arem, BV, Driel, CJGV, Visser, R. The impact of cooperative adaptive cruise control on traffic-flow characteristics. IEEE Trans Intell Transp Syst 2006;7;429–36.

[86] Santiago, I, Jiménez, Á, Vara, JM, Castro, VD, Bollati, VA, Marcos, E. Model-driven engineering as a new landscape for traceability management: a systematic literature review, Inf Softw Technol 2012;54;1340–56.

[87] Kruchten, P. The 4+1 view model of architecture, IEEE Softw 1995;12;42–50.

[88] Kulkarni, V, Reddy, S. Separation of concerns in model-driven development. IEEE Softw 2003;20;64–9.

[89] Walewski, JW, Heiles, J. Using the view model to contextualize and explain system-of-systems architecture models. In 2016 11th System of Systems Engineering Conference (SoSE), Kongsberg, Norway; 2016, p. 1–6.

[90] Perry, DE, Wolf, AL. Foundations for the study of software architecture. SIGSOFT Softw. Eng. Notes. 1992;17;40–52.

[91] Mens, T, Van Gorp, P. A taxonomy of model transformation. Electron Notes Theor Comput Sci 2006;152;125–42.

[92] ISO/IEC/IEEE systems and software engineering - architecture description. ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000) 2011;1–46.

[93] ISO/IEC international standard 19508: Information technology – Object Managment Group Meta Object Facility (MOF) Core 2014. https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/18/61844.html.

[94] Rolland, C, Achour, CB, Cauvet, C, Ralyté, J. ,Sutcliffe, A, Maiden, N, Jarke, M, Haumer, P, Pohl, K, Dubois, E, Heymans, P. A proposal for a scenario classification framework. Requir Eng 1998;3;23–47.

[95] Wiegers, KE. Peer reviews in software: a practical guide. Addison-Wesley; 2002.

[96] Weber, M, Weisbrod, J. Requirements engineering in automotive development experiences and challenges. In 10th Anniversary IEEE Joint International Conference on Requirements Engineering (RE 2002). Essen, Germany: IEEE Computer Society; 2002.

[97] Hélouët, L, Maigat, PL, Pierre, LH. Decomposition of message sequence charts. 2000. https://www.researchgate.net/publication/221030043_Decomposition_of_Message_Sequence_Charts.

[98] Davies, I, Green, P, Rosemann, M, Indulska, M, Gallo, S. How do practitioners use conceptual modeling in practice? Data Knowl Eng 2006;58;358–80.

[99] Lubars, M, Potts, C, Richter, C. A review of the state of the practice in requirements modeling. In Proceedings of the IEEE International Symposium on Requirements Engineering, San Diego, CA, USA; 1993, p. 2–14.

[100] Sikora, E, Tenbergen, B, Pohl, K. Industry needs and research directions in requirements engineering for embedded systems. Requir Eng 2012;17;57–78.

[101] Ehrig, K, Küster, JM, Taentzer, G. Generating instance models from meta models. Softw Syst Model 2009;8;479–500.

[102] Kirsopp, C, Roper, M, Haworth, B, Shepperd, M, Webster, S. Towards the development of adequacy criteria for object-oriented systems proc. EuroS-TAR'97, Edinburgh, UK, 1997. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.49.1122&rep=rep1&type=pdf.

[103] Potts, C. Using schematic scenarios to understand user needs. In Proceedings of the 1st Conference on Designing Interactive Systems: Processes, Practices, Methods, & Techniques (DIS '95), New York, NY, USA; ACM: 1995, p. 247–56.

[104] Guizzardi, G, Wagner, G, Guarino, N, van Sinderen, M. An ontologically well-founded profile for UML conceptual models. In: Persson, A, Stirna, J, editors. Advanced information systems engineering, Lecture notes in computer science, Berlin, Heidelberg, Germany: Springer; 2004, p. 112–26.

[105] Madala, K, Do, H, Aceituna, D. A knowledge acquisition approach for off-nominal behaviors. In 2018 4th International Workshop on Requirements Engineering for Self-Adaptive, Collaborative, and Cyber Physical Systems (RESACS), Banff, Canada; 2018, p. 36–43.

[106] Schirner, G, Erdogmus, D, Chowdhury, K, Padir, T. The future of human-in-the-loop cyber-physical systems. Computer 2013;46;36–45.

[107] Hurfin, M, Raynal, M. A simple and fast asynchronous consensus protocol based on a weak failure detector. Distrib Comput 1999;12;209–23.

[108] Venkatesh, V, Bala, H. Technology acceptance model 3 and a research agenda on interventions. Decis Sci 2008;39;273–315.

[109] Field, AP, Miles, J, Field, Z. Discovering statistics using R. London, England and Thousand Oaks, CA, USA: Sage; 2012. http://repository.fue.edu.eg/xmlui/handle/123456789/2902.

[110] Ricca, F, Penta, MD, Torchiano, M, Tonella, P, Ceccato, M. The role of experience and ability in comprehension tasks supported by UML stereotypes. In 29th International Conference on Software Engineering (ICSE 2007), Minneapolis, MN, USA; 2007, p. 375–84.

[111] US Federal Aviation Administration, Introduction to TCAS II, version 7.1, Washington, DC, USA; 2011. http://goo.gl/EPCYzI.