A Semi-Automated Approach to Foster the Validation of Collaborative Networks of Cyber-Physical Systems

Marian Daun, Jennifer Brings, Thorsten Weyer
paluno – The Ruhr Institute for Software Technology
University of Duisburg-Essen
45127 Essen, Germany
{marian.daun, jennifer.brings, thorsten.weyer}@paluno.uni-due.de

ABSTRACT

Cyber-physical systems form collaborative networks dynamically at runtime. In the collaboration of multiple systems, behavior emerges in the interplay of the collaborating instances. This emergent behavior raises challenges for the validation of cyber-physical systems' software, since interoperability of the single systems as well as functional correctness of the entire network of collaborative cyber-physical systems must be validated for all possible configurations of the network. Such network configurations differ, among others, in the number of participating systems, the number of system types involved, and the communication patterns between the participating systems. To aid the validation of behavior emerging from the collaboration, this paper proposes the automated generation of dedicated review diagrams to investigate the collaborative network's behavior for different network configurations. First evaluations using case examples from industry partners show that the use of such automatically generated instance level review diagrams can support the validation of collaborative cyber-physical systems.

CCS CONCEPTS

• Software and its engineering →• Software and its engineering → Model-driven software engineering; **• Software and its engineering → Software verification and validation**; **• Computer systems organization → Embedded and cyber-physical systems**

KEYWORDS

Emergent Behavior; Cyber-physical Systems; Collaborative Networks; Message Sequence Charts; Validation

SEsCPS'18, May 27-June 3 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5728-9/18/05…$15.00
https://doi.org/10.1145/3196478.3196483

1 INTRODUCTION

Cyber-physical systems are often part of a wider collaborative network consisting of many other systems. These networks typically form dynamically during run-time. Hence, these networks are continually reshaped as cyber-physical systems join and leave the network. An avionic collision avoidance system, for example, connects with other avionic collision avoidance systems in nearby aircraft, to exchange flight data. When an aircraft enters the vicinity of other aircraft, its collision avoidance system joins the network. Meanwhile other collision avoidance systems disconnect from the network when the respective aircraft leave the vicinity.

A major challenge for collaborative networks of cyber-physical systems is the validation of behavior emerging from the collaboration of the networked systems. This emergent behavior can either be a desired property, which enables some kind of functionality (like a constant speed maintained by all vehicles in a convoy to prevent a congestion) or undesired and potentially dangerous (like a rerouting of several aircraft to prevent one collision that leads to one or more new collision threats). While fully automated verification techniques can aid the validation of well-known behaviors (e.g., [1], [2]), fully automated techniques need a reference artifact which is to be considered correct (i.e. a total specification of the allowed and prohibited emerging behavior). In industrial practice, however, not every desired emergent behavior is explicitly documented and undesired emergent behavior is often forgotten [3]. Thus, early validation activities typically rely on manual reviews.

In previous work (cf. [4], [5]), we have proposed a semi-automated validation approach that uses automatically generated dedicated review models in the notation of ITU-message sequence charts [6] to aid the manual review of requirements and design artifacts of embedded systems. The proposed approach was evaluated as significantly more effective, efficient, supportive, and reviewer confidence increasing compared to traditional manual reviews of embedded systems' specifications [5]. Hence, we base our solution concept for the validation of emergent behavior from collaborative networks on these review models in ITU message sequence chart notation. Regarding the review of emerging behavior in collaborative networks, two major challenges exist [7]. First, the review artifact must allow for validating different network configurations, i.e. different numbers of participating system instances and their possible interactions must be considered.

Second, as the number of system instances within such networks may become infinite and the possible combinations of interactions between these system instances increases exponentially with each additional instance, the review artifact must not only enable the review of all possible network configurations, but at the same time limit the review effort to a reasonable amount.

To address these challenges, this paper contributes to the review of emergent behavior in networks of cyber-physical systems by proposing the automated generation of instance models which allow the review of different network configurations. The approach relies on the automated generation of diagrams to validate different network configurations considering multiple instances of the participating systems. In addition, we propose the detection of parts in different configuration diagrams, which exhibit the identical overall behavior, to limit the number of diagrams to be reviewed.

This paper is structured as follows: Section 2 summarizes the related work. Subsequently, Section 3 will introduce our solution approach to generate review diagrams for different network configurations considering multiple instances of the same system type. Section 4 then discusses our initial findings from industrial evaluation of the proposed approach and finally, Section 5 concludes the paper.

2 RELATED WORK

With respect to the problem situation and to our solution approach lots of research has been done in different research areas. Hence, this section briefly summarizes some approaches closely related to collaborative networks of cyber-physical systems. In particular, approaches dealing with the validation of collaborative networks and the analysis of emergent behavior as well as approaches dealing with the issue of unknown network configurations and multiple participating instances of the same type must be considered relevant. In addition, works dealing with the analysis of message sequence charts with respect to implicit scenarios and emerging behavior is related to the proposed approach.

2.1 Validation of Collaborative Networks and Analysis of Emergent Behavior

There exist a plethora of approaches to foster automated verification or analysis of emergent behavior. Approaches are typically widely recognized and have been proposed years ago. These approaches seem in principle transferable to the verification and analysis of collaborative networks of cyber-physical systems. For instance, approaches have been proposed to aid automated verification of systems designed for such situations (e.g., [1], [2]). In addition, approaches exist that focus on the detection of emergent behavior such as [8]. Another branch of research addressing the verification of such systems uses simulation to analyze the collaborative network (e.g., [9], [10]). Further approaches have been proposed in the field of multi-agent systems, where many individual systems are interconnected in order to achieve a common goal [11]. Specifically approaches coping with non-governed networks (i.e. where no central control logic exists) are of interest (cf. [12]). A major research area in this field of expertise is the area of machine learning techniques [13], which can be used to estimate the optimal behavior for given context stimuli.

Another related field is the area of feature interactions in embedded systems (cf. [14]). As the collaborative network can be seen as a system on its own, approaches coping with the feature interaction problem (e.g., [15], [16]) can be used to detect and prevent occurrence of emergent behavior. Furthermore, related approaches for verification and validation of swarm systems have been proposed (cf. [17], [18]). These approaches can typically deal with an undefined or even an infinite number of participating systems.

While all these approaches are in principle transferable to the problem domain, we did not follow this train of thought. As outlined in Section 1, automated verification approaches have several preconditions that do not apply in our case. For instance, it is assumed that all desired and undesired behavior is known and explicitly specified. This assumption does not hold in our case.

2.2 Uncertainty in Collaborative Network Configuration and Multiple Instances

Further approaches address the problem at runtime, answering the question, how to enable systems to dynamically decide how to adapt to given context situations (e.g., [19], [20]). In particular, correct behavior must be ensured in dynamically changing collaborating networks of cyber-physical systems in the presence of uncertainty during runtime. Therefore, research deals with monitoring, adapting to, and handling of unforeseen situations at runtime. In particular, techniques for runtime monitoring and analysis can be used to cope with configurations that have not been validated at design time (cf. [21], [22]). In addition, model-based adaptation (e.g., [23], [24]) seems appropriate to aid in runtime handling of uncertain collaborative network configurations. Another approach towards coping with uncertainty in configuration makes use of model-based documentation, e.g., by explicitly documenting uncertainty regarding the number and the specific type of involved components (e.g., [25], [26]).

Regarding the multiple instance phenomenon in collaborative networks of cyber-physical, well-known approaches deal with the model-based documentation at instance level (e.g., [52], [24], [27]). Further documentation approaches propose the distinction between type and instance level models (e.g., [28]) and document instance and type level elements in integrated diagrams (e.g., [29]).

2.3 Implicit and Emergent Behavior in Message Sequence Charts

The specification of the interaction-based behavior of systems is required in the automotive and avionics domain (cf. [30], [31]). In particular, it is common and it has proven suitable to use message sequence charts for this task (cf. [49], [32]). Hence,

related approaches exist, which address the analysis of message sequence chart specifications w.r.t. potentially unsafe behavior emerging from collaboration. Two approaches are of particular importance and also build the foundations for Section 3: first, approaches dealing with implied scenarios; i.e. behavior that is not explicitly specified within a message sequence charts document but emerges from the combination of different diagrams (e.g., [33], [34], [35]) or from the causal ordering of message sequence charts (e.g., [36], [37]) and second, approaches dealing with emergent behavior; i.e. behavior resulting from the interplay of multiple systems in in a collaborative network (e.g., [38], [29]). While the related approaches focus on the detection of emergent behavior, we aim at properly displaying emergent behavior in different network configurations of the collaborative network to aid the manual review of specifications in early development phases.

3 SEMI-AUTOMATED VALIDATION OF COLLABORATIVE NETWORKS

As outlined above, in literature attention is given to the detection of emergent behavior in MSC-based behavior specifications. To this end, distinct diagrams can be generated automatically to explicitly show the emergent behavior resulting from the interplay of multiple cyber-physical systems as outlined in [7]. In particular, based on existing general works, we proposed the automated generation of dedicated MSC diagrams to depict the interplay of multiple system-instances in different configurations of collaborative networks. In this section, Section 3.1 will introduce the proposed approach and Section 3.2 will show that the proposed automated generation of review diagrams is algorithmically feasible. Finally, Section 3.3 will discuss the use of the algorithmic approach for the generation of diagrams for different network configurations. Attention is also given to the problem of scalability of the approach which results in numerous review diagrams and how this issue can be addressed in future work.

3.1 Overview on Semi-Automated Approach

To foster the validation of behavior emerging from the interplay of such collaborative networks, we propose the use of specific review diagrams, displaying the collaborative network in different configurations at instance level. Since ITU Message Sequence Charts have proven as effective, efficient and supportive for manual validation tasks (cf. [4]). The proposed approach is based on the use of message sequence charts as review artifacts.

Figure 1 introduces the proposed process steps. As precondition the approach relies on the existence of model-based MSC specifications on the type-level, which are used as input. MSCs or MSC-like languages are commonly used for scenario specifications and for detailed specification of the intended interaction-based system of behavior. The approach consists of the automated generation of dedicated review diagrams (Steps 1 and 2) and on the manual review of these diagrams (Step 3). In more detail, the three steps are:

- Step 1: In the first automated step the specification of the system under review, which is commonly specified on the type level, is used as input to generate instance level review diagrams. For each bMSC diagram of the specification an instance level review diagram is generated. In contrast to the original specification, these review diagrams display details of context entities on the same level of abstraction as the system itself is displayed in the original specification.
- Step 2: In the second automated step, the instance level diagrams from Step 1 are taken as input to increase the number of involved system instances. Additionally, according to different network configurations different possible communication patterns between the system instances are calculated. Review diagrams generated in such a way allow for the investigation of behavior emerging in different network configurations.
- Step 3: Finally, the generated review diagrams are manually reviewed by experts. In addition, we assume that also the original specification at the type level should be reviewed as not only collaborative properties but also general single system properties need to be considered.

This section elaborates on the automated generation of specific review diagrams to allow for manual validation of behavior emerging from collaborations. To illustrate our approach, we will use simplified sketches of a collaborative adaptive cruise control system, which exchanges its current velocity with other adaptive cruise control systems. Such an exchange is, for example, necessary to dynamically form convoys in traffic jam situations. Next, Section 3.1 introduces the foundations of the used notation format and shows the generation of instance level review diagrams as proposed in Step 1. Subsequently, Section 3.2 discusses the generation of instance level review diagrams for different network configurations. In doing so, emphasis is given to limiting the number of configuration diagrams to be reviewed to a reasonable amount.

3.2 Automated Generation of Instance Level Review Diagrams

Interaction-based specifications often use the MSC notation. A MSC-based specification consists of high-level Message Sequence Charts (hMSC), to logically structure the execution paths, and basic Message Sequence Charts (bMSC), which define the interaction-based behavior. Commonly, a bMSC b is defined as a 5-tupel $b = \langle I_b, A_b, E_b, \leq_b, \alpha_b \rangle$ (cf. e.g., [39]), where:

- I is a set of bMSC-instances representing the system itself or context entities the system interacts with,
- A is a set of messages exchanged between the bMSC-instances,
- E is a set of events. An event represents a discrete point in time when bMSC-instances send or receive messages,
- $\leq$ is a partial order relation, timely ordering events, and
- $\alpha \subseteq E \times A \times I$ is a relation, defining which event is related to which message and to which bMSC-instance.

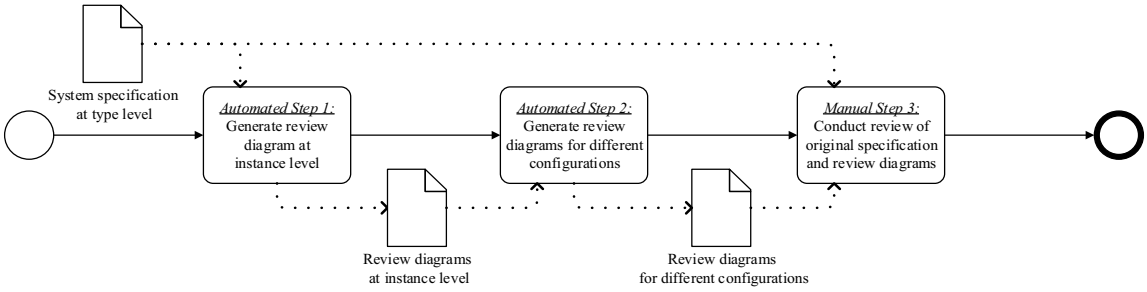

Figure 1: Validation of collaborative networks using automatically generated instance level review diagrams

In the engineering of cyber-physical systems abstraction mechanisms are commonly used to cope with the increasing complexity as abstractions allow to investigate the same situation on different levels of granularity (cf. [40]). Hence, we can further define:

- $i^{\mathbf{s}} \in I$ as the bMSC-instance representing the system **s**,
- $I^{\mathbf{s}}$ as a set of bMSC-instances representing the components of the s system **s**, and
- $I^{\mathbf{c}}$ as a set of bMSC-instances representing context entities.

In case of MSCs, a composition operator for bMSC-instances allows to compose the system's bMSC-instance from bMSC-instances representing the system's components. Accordingly, two bMSC diagrams display the same behavior on different levels of abstraction if (a) the context entities are identical, (b) the message exchange with the context is identical, and (c) one bMSC displays $i^{\mathbf{s}}$ and the other bMSC displays a subset of $I^{\mathbf{s}}$. Therefore, the bMSC-instances of a single bMSC are either $i^{\mathbf{s}}$ and a subset of $I^{\mathbf{c}}$ (which we will refer to as $I_b^{\mathbf{c}}$) or $I_b \subseteq I^{\mathbf{s}} \cup I^{\mathbf{c}}$ (for the relevant subset of $I^{\mathbf{s}}$, we will accordingly refer to as $I_b^{\mathbf{s}}$). Figure 2 sketches this situation.

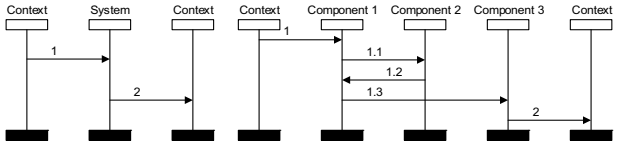

Figure 2: Identical behavior on different abstraction levels

For collaborative networks of cyber-physical a context entity can be another system of the same type as defined in detail within the same bMSC. For example, while one instance of an adaptive cruise control (ACC) system is specified by $\langle I_b^{\mathbf{s}} \cup I_b^{\mathbf{c}}, A_b, E_b, \leq_b, \alpha_b \rangle$, the ACC system is not only represented by $i^{\mathbf{s}}$ or $I_b^{\mathbf{s}}$ but also by one bMSC-instance $i^{\mathbf{c}}$ of $I_b^{\mathbf{c}}$, which represents another instance of the same type of ACC system in the context of the system under specification. Therefore, it holds that $\exists (i^{\mathbf{c}} \in I^C | i^{\mathbf{c}} = i^{\mathbf{s}})$. Hence, the specification must consist of a bMSC b' detailing the system behavior in terms of inverse message in- and outputs. This means: $\exists b'$ where $\exists i^{\mathbf{c}} \in I_b^{\mathbf{s}} \wedge (i^{\mathbf{c}} \in I^C | i^{\mathbf{c}} = i^{\mathbf{s}})$ and it holds that the inputs to $i^{\mathbf{c}}$ in b' are identical to the inputs to $i^{\mathbf{s}}$ in b, the outputs from $i^{\mathbf{c}}$ in b' are identical to the outputs from $i^{\mathbf{s}}$ in b, and these in- and outputs are ordered the same in b' and b.

In many cases the same bMSC will define original and inverse behaviors, as the systems are typically designed to exchange the same kind of information between each other. For example, Figure 3 illustrates this situation. Two ACC systems exchange their current speed, which will later be needed, e.g., to calculate and ensure a common convoy velocity or to avoid rear-end collisions. Hence, in the development artifacts, it is specified that the system under development (i.e. the ACC marked as system, which consists of the components ACC Control Unit and ACC Interface) communicates with another ACC in its context. This ACC, of course, once again is another representative of the system under development.

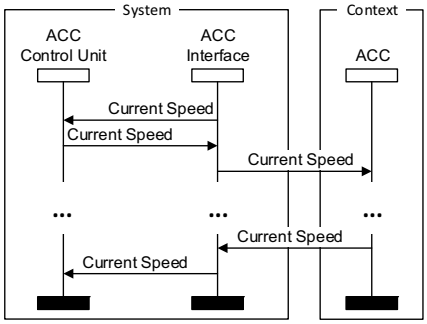

Figure 3: Identical behavior of system and context instances.

As a first step towards a proper review of the behavior emerging from collaborative networks of cyber-physical systems, it is important not only to review the behavior emerging from the interplay with abstract context entities representing a specific type of system but to review the behavior emerging from the interplay of the participating instances of the same system type [7]. Hence, we suggest to refine the given bMSCs' context entities with information about their internal structure and behavior. In case of multiple communicating instances of the same type, such a refined diagram can be generated automatically. In order to do so, the context instances that represent another instance of the same system must be identified and replaced by copies of the system instances. These copies now represent another system of the same type in the context of the original system.

This is illustrated for two system instances of the same type in Figure 4. Figure 4 shows the same situation as depicted in Figure 3 but the black box view on the ACC in the context has

been replaced with the information defined for the ACC under development. Hence, the internal behavior of both ACCs needed for proper collaboration can be investigated by a reviewer.

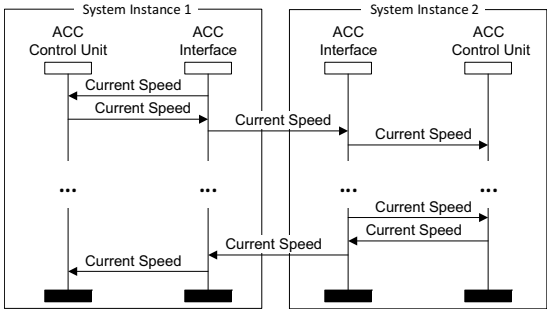

Figure 4: Collaboration between two system instances

For the situation depicted in Figure 3, where the communication can be considered inverse, this means that a refined bMSC $b^{\mathbf{ref}} = \langle I^{\mathbf{s}}_{\mathbf{ref}} \cup I^{\mathbf{c}}_{\mathbf{ref}}, A_{\mathbf{ref}}, E_{\mathbf{ref}}, \leq_{\mathbf{ref}}, \alpha_{\mathbf{ref}} \rangle$ can be derived by transformation from $b = \langle I^{\mathbf{s}}_b \cup I^{\mathbf{c}}_b, A_b, E_b, \leq_b, \alpha_b \rangle$ as follows:

- The set of bMSC-instances represents all system or component instances of b, all context instances of b representing systems other than the system itself and copies of the system instances in lieu of the context instance representing another instance of the system:

$$I^{\mathbf{s}}_{\mathbf{ref}} = I^{\mathbf{s}}_b$$
$$I^{\mathbf{c}}_{\mathbf{ref}} = (I^{\mathbf{c}}_b \setminus i^{\mathbf{c}}) \cup \{i' | \forall i \in I^{\mathbf{s}}_b \rightarrow \exists i' \in I^{\mathbf{c}}_{\mathbf{ref}}\}$$

- The set of messages contains all external messages, all internal messages and a copy of these internal messages for the second instance of the system under development:

$$A_{\mathbf{ref}} = A_b \cup \left\{ a' \middle| \forall \left(a \in A_b \middle| \begin{matrix} \exists (e_1, a, i_1), \\ (e_2, a, i_2) \in \alpha_b \\ \wedge\, i_1, i_2 \in I^{\mathbf{s}}_b \end{matrix} \right) \rightarrow \exists a' \in A_{\mathbf{ref}} \right\}$$

- The set of events contains all events related to external messages (with the exception of events related to the second system as context instance), all events related to internal messages for the first system and a copy of these events related to internal messages for the second system instance:

$$E_{\mathbf{ref}} = (E_b \setminus \{e | \exists (e, a, i^{\mathbf{c}}) \in \alpha_b\}) \cup \{e' | \forall (e \in E_b | \exists (e, a, i) \in \alpha_b \wedge i \in I^{\mathbf{s}}_b) \rightarrow \exists e' \in E_{\mathbf{ref}}\}$$

- Regarding the ordering, the events are ordered as in b (this also holds for the copied events).
- The messages sent or received by the second system are related to the corresponding instances:

$$\forall (e, a, i^{\mathbf{c}}) \in \alpha_b \rightarrow \exists \big((e', a, i') \in \alpha_{\mathbf{ref}} | i \rightarrow i' \wedge \exists (e_n, a, i) \in \alpha_b \big)$$

3.3 Generation of Review Diagrams for Different Network Configurations

Diagrams for network configurations with more than two instances of the same type can be generated similarly. New system instances can be added successively either as one bMSC instance representing one system instance or as several bMSC instances which also display the internal behavior of the new system instance. Figure 5 shows a bMSC diagram for a network of four ACCs. This allows reviewing and discussing the intended behavior of the collaborative network in case multiple connections exist. For example, a reviewer will have to decide whether the order of received and sent messages for ACC4 is important (e.g., the car's speed is adapted to the first incoming speed message of another ACC, and hence the second incoming message might overwrite this value, resulting in unsafe behavior) or negligible (e.g., the current speed data is exchanged but does not impact the internal behavior of the ACC).

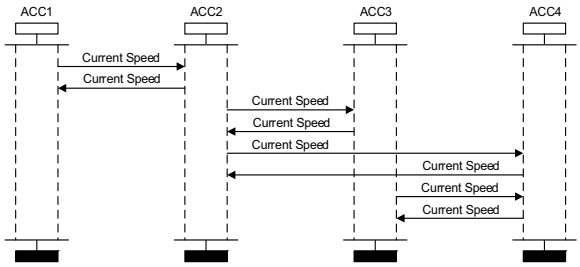

Figure 5: Multiple instance diagram

The automated generation of diagrams for different configurations of the collaborative network allows in principle for considering all possible configurations. However, the review of all configurations is infeasible, as the number of network configurations may become infinite. For example, when adding another system instance to a collaborative network several possibilities for communication between the new system instance and the system instances already in the network arise. The new instance could for example communicate with all other system instances in the network (complete communication) or just with one (simple communication). For collaborative networks consisting of more than three system instances, several other combinations exist, i.e. the new system instances could communicate with several other system instances in the network but not with all.

To cope with the increasing number of review diagrams, we propose two solution concepts. First of all, due to the manual nature of the review process it might be acceptable to limit the review to a certain number of collaborating identical systems within a collaborative network. For instance, in a platoon driving on a highway, in fact, there will be a maximum number of systems collaborating. Another idea is the identification of identical behavior in different configurations. These identified identical parts should then only be reviewed once. This is based on the idea that not all configurations lead to completely new behavior of the collaborative network. Hence, configurations must be identified that do not reveal additional insight into the

behavior emerging from the collaborative network. For example, Figure 6 shows two different network configurations. While the number of participating system instances differs, the communication patterns are identical in both network configurations. Each system instances either exchanges information with one other system instance or with two other system instances.

Both situations can be investigated in distinct diagrams to analyze the impact of the system collaboration in the different network configurations. Figure 7 sketches such a diagram displaying the internal behavior for the interaction of two system instance communicating with one other system instance. In addition, Figure 8 sketches the diagram displaying the internal behavior for the interaction of three instances.

The identification of such recurring interactions within collaborative networks of different sizes allows to automatically decrease the number of configurations, which must be considered during a review. In case interacting system instances behave like a single system instance itself, composition can be used to reduce the complexity of the configuration under investigation. In doing so, many configurations can be composed to a state where this configuration is identical to another configuration already depicted in review diagrams.

For a review diagram displaying one configuration this means, if (a) there exists a set of bMSC-instances I^{CP} where each instance $i^{CP} \in I^{CP}$ represents an instance of the system $\boldsymbol{s}$, and (b) I^{CP} exhibits the same behavior as a single system instance $i^{\boldsymbol{s}}$, I^{CP} represents a recurring part of a communication pattern and can hence be replaced with a single system instance $i^{\boldsymbol{s}}$.

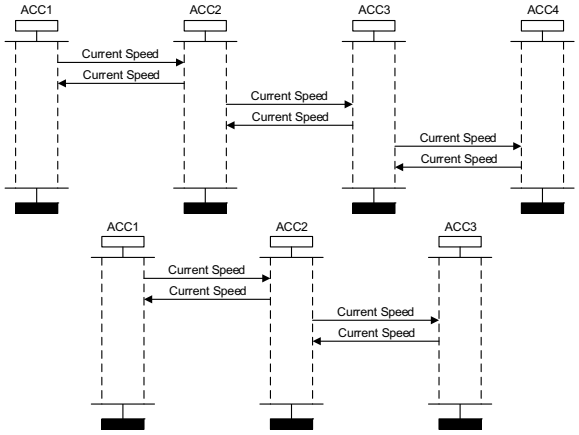

Figure 6: Two network configurations with the same communication patterns

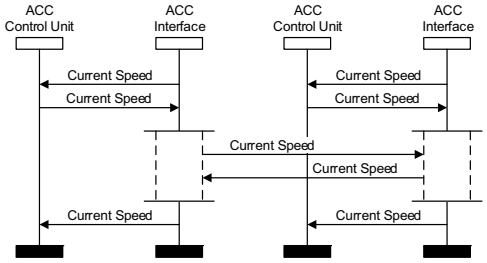

Figure 7: Review diagram for two communicating systems

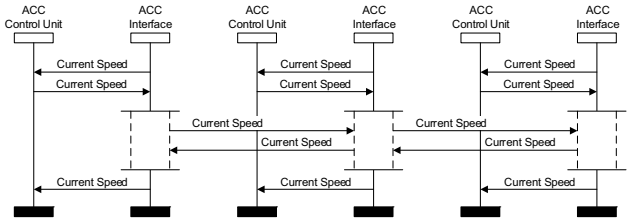

Figure 8: Review diagram for three communicating systems

4 INITIAL EVALUATION

As a proof of concept, we applied the approach to an industrial case example from the avionics domain and discussed the resulting diagrams as well as the approach itself with our industrial partners. The case example was provided by industry: An advanced collision avoidance system collaborates with other collision avoidance systems in the vicinity of the aircraft. In contrast to common collision avoidance systems, the aircraft exchange their flight data and adapt their flight paths as early as possible. This allows for a more comfortable flight, as it avoids rapid changes in altitude as is necessary when handled by common collision avoidance systems.

The early exchange of flight paths and the adaption of the route lead to multiple instances of these advanced collision avoidance systems communicating with each other and, hence, potential for possible defects. For instance, if two aircraft negotiate new routes to avoid a collision, it is important that this route is not changed back when negotiating the routes with the next aircraft entering vicinity.

Our initial evaluation shows that the approach is applicable to the case example. Industry professionals substantiated our impression of usefulness of this approach to support manual validation of cyber-physical systems, which are heavily collaborative. Particularly, it showed off that, as intended, different review diagrams (i.e. diagrams showing a different number of instances handling the same situation) is beneficial to identify potential safety defects. Industry experts stressed the usefulness of investigating concrete scenario-like instance configurations instead of type-level diagrams. However, we found that different diagrams provide different insights, which does not only hold for instance vs type-level diagrams but, furthermore, also to different type-level diagrams (i.e. diagrams with two instances vs diagrams with three instances). Hence, future work will have to deal with investigating, which instance level configurations are advantageous for manual reviews.

5 CONCLUSIONS

In this paper, we have discussed the need to validate behavior emerging from the collaboration of cyber-physical systems. This emergent behavior can be specific to a configuration, leading to the need to investigate all possible configurations to ensure functional suitability of the entire network and correct interoperability of each participating system. Thus, the problem

of infinite or nearly infinite numbers of configurations of collaborative networks arises. To provide support for manual reviews of emergent behavioral properties, we presented an approach which relies on the generation of review diagrams to allow for inspection of different configurations of collaborative networks. To cope with infinite possibilities of network configurations, the approach proposes the detection of identical parts in different collaborative network configurations. This allows to validate emergent properties and to reduce the complexity of the total collaborative network configuration under review. Furthermore, the solution concept aids in limiting the number of diagrams, which need to be reviewed. First evaluation results are promising. Particularly, insights from the application of the approach to an industrial case example and discussions with industry partners showed that the generation and the manual review of instance-level review diagrams can support the early detection of safety defects.

ACKNOWLEDGMENTS

This research was funded by the *German Federal Ministry of Education and Research* under grant no. 01IS12005C and grant no. 01IS16043V. We like to thank our industrial partners Stefan Beck and Arnaud Boyer (Airbus) for their support in application to the case example and the fruitful discussions.

REFERENCES

[1] B. Becker, D. Beyer, H. Giese, F. Klein, and D. Schilling, 'Symbolic Invariant Verification for Systems with Dynamic Structural Adaptation', in *Proceedings of the 28th International Conference on Software Engineering*, New York, NY, USA, 2006, pp. 72–81.

[2] H. Giese, S. Burmester, W. Schäfer, and O. Oberschelp, 'Modular Design and Verification of Component-based Mechatronic Systems with Online-reconfiguration', in *Proceedings of the 12th ACM SIGSOFT Twelfth International Symposium on Foundations of Software Engineering*, New York, NY, USA, 2004, pp. 179–188.

[3] M. Daun, J. Höfflinger, and T. Weyer, 'Function-centered Engineering of Embedded Systems - Evaluating Industry Needs and Possible Solutions', in *ENASE 2014 - Proceedings of the 9th International Conference on Evaluation of Novel Approaches to Software Engineering, Lisbon, Portugal, 28-30 April, 2014*, 2014, pp. 226–234.

[4] M. Daun, T. Weyer, and K. Pohl, 'Detecting and Correcting Outdated Requirements in Function-Centered Engineering of Embedded Systems', in *Requirements Engineering: Foundation for Software Quality*, S. A. Fricker and K. Schneider, Eds. Springer International Publishing, 2015, pp. 65–80.

[5] M. Daun, A. Salmon, T. Weyer, and K. Pohl, '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, April 27-29, 2015*, 2015, p. 29:1–29:6.

[6] International Telecommunication Union, 'User Requirements Notation (URN)', Z 151, 2012.

[7] M. Daun, J. Brings, T. Bandyszak, P. Bohn, and T. Weyer, 'Collaborating Multiple System Instances of Smart Cyber-physical Systems: A Problem Situation, Solution Idea, and Remaining Research Challenges', in *1st IEEE/ACM International Workshop on Software Engineering for Smart Cyber-Physical Systems, SEsCPS 2015, Florence, Italy, May 17, 2015*, 2015, pp. 48–51.

[8] S. Malakuti, 'Detecting Emergent Interference in Integration of Multiple Self-Adaptive Systems', in *Proceedings of the 2014 European Conference on Software Architecture Workshops*, New York, NY, USA, 2014, p. 24:1–24:7.

[9] B. Bauer, J. P. Müller, and J. Odell, 'Agent UML: A Formalism for Specifying Multiagent Software Systems', in *Agent-Oriented Software Engineering*, vol. 1957, G. Goos, J. Hartmanis, J. van Leeuwen, P. Ciancarini, and M. J. Wooldridge, Eds. Berlin, Heidelberg: Springer, 2001, pp. 91–103.

[10] F. Klein and H. Giese, 'Analysis and design of physical and social contexts in multi-agent systems using UML', *ACM SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, pp. 1–8, 2005.

[11] W. van der Hoek and M. Wooldridge, 'Chapter 24 Multi-Agent Systems', in *Foundations of Artificial Intelligence*, vol. 3, V. L. and B. P. Frank van Harmelen, Ed. Elsevier, 2008, pp. 887–928.

[12] J. Ferber, *Multi-Agent System: An Introduction to Distributed Artificial Intelligence*, vol. 4. Addison Wesley, 1999.

[13] T. M. Mitchell, *Machine Learning*, 1st ed. New York: McGraw-Hill, 1997.

[14] T. F. Bowen, F. S. Dworack, C.-H. Chow, N. Griffeth, G. E. Herman, and Y.-J. Lin, 'The feature interaction problem in telecommunications systems', in *Int. Conf. on SE for Telecommunication Switching Systems*, 1989, pp. 59–62.

[15] K. Kimbler and L. G. Bouma, Eds., *Feature interactions in telecommunications and software systems V*. Amsterdam; Washington, DC: Tokyo, Japan: IOS Press; Ohmsha, 1998.

[16] A. P. Felty and K. S. Namjoshi, 'Feature Specification and Automated Conflict Detection', *ACM Trans Softw Eng Methodol*, vol. 12, no. 1, pp. 3–27, Jan. 2003.

[17] P. Kouvaros and A. Lomuscio, 'A counter abstraction technique for the verification of robot swarms', in *29th AAAI Conference on Artificial Intelligence, AAAI 2015*, 2015, vol. 3, pp. 2081–2088.

[18] P. Kouvaros and A. Lomuscio, 'Verifying emergent properties of swarms', in *24th International Joint Conference on Artificial Intelligence, IJCAI 2015*, 2015, vol. 2015–January, pp. 1083–1089.

[19] I. Georgiadis, J. Magee, and J. Kramer, 'Self-organising Software Architectures for Distributed Systems', in *1st Workshop on Self-healing Systems*, 2002, pp. 33–38.

[20] A. Rook, A. Knauss, D. Damian, H. A. Muller, and A. Thomo, 'Integrating Data Mining into Feedback Loops for Predictive Context Adaptation', Victoria BC, DCS-349-IR, 2013.

[21] J. O. Kephart and D. M. Chess, 'The vision of autonomic computing', *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[22] J. Mao and L. Chen, 'Runtime Monitoring for Cyber-physical Systems: A Case Study of Cooperative Adaptive Cruise Control', in *2nd International Conference on Intelligent System Design and Engineering Application*, 2012, pp. 509–515.

[23] J. C. Georgas, A. Van der Hoek, and R. N. Taylor, 'Using Architectural Models to Manage and Visualize Runtime Adaptation', *Computer*, vol. 42, no. 10, pp. 52–60, Oct. 2009.

[24] J. Floch, S. Hallsteinsen, E. Stav, F. Eliassen, K. Lund, and E. Gjorven, 'Using architecture models for runtime adaptability', *IEEE Softw.*, vol. 23, no. 2, pp. 62–70, 2006.

[25] P. Vařeková, B. Zimmerova, P. Moravec, and I. Černá, 'Formal verification of systems with an unlimited number of components', *IET Softw.*, vol. 2, no. 6, pp. 532–546, 2008.

[26] M. Blay-Fornarino, A. Charfi, D. Emsellem, A.-M. Pinna-Déry, and M. Riveill, 'Software interactions', *J Object Technol.*, vol. 3, no. 10, pp. 161–180, 2004.

[27] S. Alda, M. Won, and A. B. Cremers, 'Managing Dependencies in Component-Based Distributed Applications', in *Scientific Engineering for Distributed Java Applications*, vol. 2604, G. Goos, J. Hartmanis, J. van Leeuwen, N. Guelfi, E. Astesiano, and G. Reggio, Eds. Berlin, Heidelberg: Springer, 2003, pp. 143–154.

[28] C. Rolland *et al.*, 'A proposal for a scenario classification framework', *Requir. Eng.*, vol. 3, no. 1, pp. 23–47, 1998.

[29] F. H. Fard and B. H. Far, 'Detecting emergent behavior in autonomous distributed systems with many components of the same type', in *Int. Conf. on Systems, Man and Cybernetics*, 2012, pp. 1924–1929.

[30] ISO, 'ISO 26262-1:2011 Road vehicles -- Functional safety', 2011.

[31] D. Lempia and S. Miller, 'Requirements Engineering Management Handbook', Federal Aviation Administration, DOT/FAA/AR-08/32, 2009.

[32] S. Mauw, M. A. Reiniers, and T. A. C. Willemse, 'Message Sequence Charts in the Software Engineering Process', in *Handbook of Software Engineering & Knowledge Engineering: Fundamentals*, World Scientific, 2002, pp. 437–464.

[33] S. Uchitel, J. Kramer, and J. Magee, 'Detecting implied scenarios in message sequence chart specifications', in *Proc. of ESEC*, 2001, pp. 74–82.

[34] E. Letier, J. Kramer, J. Magee, and S. Uchitel, 'Monitoring and control in scenario-based requirements analysis', in *27th International Conference on Software Engineering, 2005. ICSE 2005. Proceedings*, 2005, pp. 382–391.

[35] R. Alur, K. Etessami, and M. Yannakakis, 'Inference of message sequence charts', *IEEE Trans. Softw. Eng.*, vol. 29, no. 7, pp. 623–633, Jul. 2003.

[36] C.-A. Chen, S. Kalvala, and J. Sinclair, 'Race Conditions in Message Sequence Charts', in *Programming Languages and Systems*, vol. 3780, K. Yi, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 195–211.

[37] B. Mitchell, 'Inherent Causal Orderings of Partial Order Scenarios', in *Theoretical Aspects of Computing - ICTAC 2004*, vol. 3407, Z. Liu and K. Araki, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 113–127.

[38] M. Moshirpour, A. Mousavi, and B. H. Far, 'Detecting Emergent Behavior in Distributed Systems Using Scenario-Based Specifications', *Int. J. Softw. Eng. Knowl. Eng.*, vol. 22, no. 06, pp. 729–746, 2012.

[39] L. Hélouët and P. L. Maigat, 'Decomposition of Message Sequence Charts', in *SAM 2000, 2nd Workshop on SDL and MSC, Col de Porte, Grenoble, France, June 26-28, 2000*, 2000, pp. 47–60.

[40] N. G. Leveson, 'Intent specifications: an approach to building human-centered specifications', in *1998 Third International Conference on Requirements Engineering, 1998. Proceedings*, 1998, pp. 204–213.