2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems

Component and Connector Views in Practice: An Experience Report

Vincent Bertram[1], Shahar Maoz[2], Jan Oliver Ringert[2], Bernhard Rumpe[3], Michael von Wenckstern[3]
[1] Daimler AG Group Research & MBC Development, Ulm, Germany
[2] School of Computer Science, Tel Aviv University, Israel
[3] RWTH Aachen University, Aachen, Germany

***Abstract*—Component and Connector (C&C) view specifications, with corresponding verification and synthesis techniques, have been recently suggested as a means for formal yet intuitive structural specification of C&C models. In this paper we report on our recent experience in applying C&C views in industrial practice, where we aimed to answer questions such as: could C&C views be practically used in industry, what are challenges of systems engineers that the use of C&C views could address, and what are some of the technical obstacles in bringing C&C views to the hands of systems engineers. We describe our experience in detail and discuss a list of lessons we have learned, including, e.g., a missing abstraction concept in C&C models and C&C views that we have identified and added to the views language and tool, that engineers can create graphical C&C views quite easily, and how verification algorithms scale on real-size industry models. Furthermore, we report on the non-negligible technical effort needed to translate Simulink block diagrams to C&C models. We make all materials mentioned and used in our experience electronically available for inspection and further research.**

***Index Terms*—component and connector models, Simulink, architecture, industrial case study**

I. Introduction

C&C models, described using languages such as SysML [28], AADL [8], [9], and related block diagram languages, are used extensively in software and systems engineering. Simulink/Stateflow [18], [19] are prevalent tools used in the automotive industry for model-based prototype implementation, simulation, and testing.

Recently, we have presented C&C views [16], as a means to formally and intuitively specify constraints on the structure of C&C models. The views allow engineers to specify constraints on hierarchy and connectivity, using partial examples, while crosscutting the implementation-oriented system/subsystem hierarchy of the target model. The verification problem of checking a C&C model against a view was investigated in [17]. The synthesis problem of automatically generating a C&C model satisfying a given C&C views specification, if one exists, was studied in [16].

While the abstractions and algorithms introduced for C&C views look interesting, they have not previously been evaluated in an industrial setting. The papers describing C&C views use both synthetic and adapted real-world models, but they focus on introducing and examining the new concepts and algorithms, not on their concrete application in practice.

In this paper we report on our experience in applying C&C views in practice, in an industrial, automotive setting, in order to answer the following four main questions:

- **Q1** Which industrial contexts in automotive domain are relevant for C&C views and what challenges can the use of C&C views address?
- **Q2** Can domain experts create C&C views with reasonable effort and are they missing any language features?
- **Q3** Is C&C views verification applicable to automotive industry models and does it scale to deal with their size?
- **Q4** Are the verification outputs of use for the engineers?

Since the answer to **Q1** influences the experiment setup for the other questions, we decided to do a two-stage study. In the preliminary study, interviewing automotive industrial partners, we investigated industrial development processes in automotive domain including data/artifacts and challenges of developers. Based on the findings of the preliminary study, the answers to **Q1**, we chose an automotive partner and relevant documents and models for evaluation. We then executed the main study, to address questions **Q2** to **Q4**.

We chose the automotive domain as representative for safety-critical, distributed control systems [21]. This choice is based on existing automotive research collaborations, the availability of requirement documents and models, no need for very domain specific expertise in order to understand the requirements, and initial feedback from domain experts.

In our main case study, two domain experts (first and last listed authors) created 50 C&C views based on 183 industrial textual requirements and design decisions of two automotive software systems: Advanced Driver Assistance Systems (ADAS), available in four different evolution versions, and Adaptive Light System (ALS). We devised a translation from Simulink block diagrams to C&C models to check the created C&C views using our existing verification tool. The translation involved non-negligible technical efforts. Finally, we presented the tool's generated witnesses, which demonstrate reasons for satisfaction or non-satisfaction, to the industrial partner who evaluated their usefulness with regard to two identified industrial challenges: traceability and evolution.

As part of our results, the industrial partner identified a missing abstraction concept in C&C views that we implemented. We found that given textual requirements, domain experts can create C&C views that highlight the implementation details of requirements in a Simulink model of hundreds of blocks with reasonable effort. We found that C&C views verification scales well for sizes of industrial models and

978-1-5386-3492-9/17 $31.00 © 2017 IEEE
DOI 10.1109/MODELS.2017.29

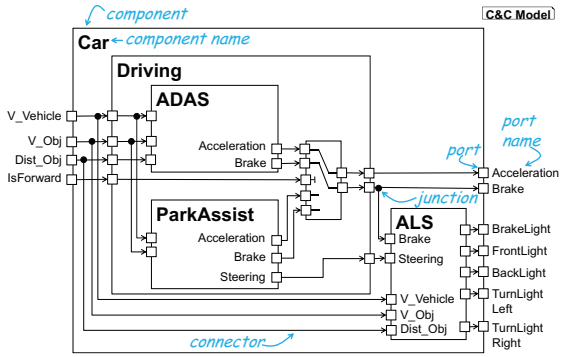

Fig. 1. Example C&C model (effectors of all atomic blocks are omitted, in this example every input port effects every output port of an atomic block with exception of the `Switch` block)

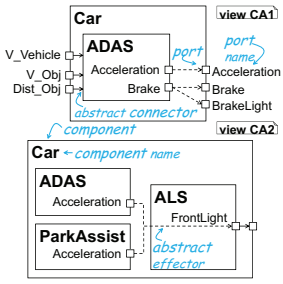

Fig. 2. Two C&C views *CA1* (top) and *CA2* (bottom)

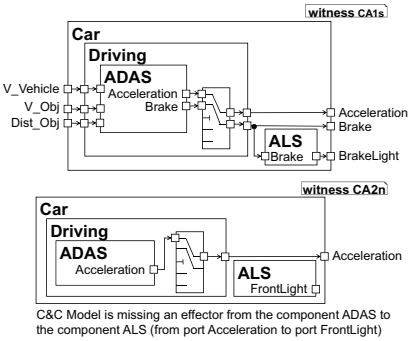

Fig. 3. Witness for satisfaction *CA1s* (top) and witness for non-satisfaction *CA2n* (bottom)

average running times were below two seconds in all our experiments. Finally, C&C views helped the domain experts to discover several inconsistencies between requirements and their implementation.

From a methodological point of view, we attempted to roughly follow the framework suggested in [24], e.g., presenting the objective, the case, the theory, the method, and the selection strategy. We followed this framework to the extent it was applicable in our context.

Finally, as an important contribution of our work we have made **all artifacts we used and created available from** [29]. These materials include the four ADAS and the one ALS Simulink models (web export) by Daimler AG, their original requirements in German with an English translation, C&C views in textual and graphical representation, and all verification results. We encourage the reader to inspect these materials and use them for their own research.

II. Background and Example

C&C models describe functional, logical or software architectures [25] in terms of components executing computations and connectors effecting component interaction via typed and directed ports. Components can be hierarchically composed from other components and components interact only via connectors. This encapsulation and logical decomposition allows efficient development, modular reuse, and evolution.

C&C views, as presented in [16], introduce four major abstraction mechanisms over hierarchy, connectivity, data flow, and interfaces of C&C models. The hierarchy of components in C&C views is not necessarily direct, abstract connectors can cross-cut component boundaries, abstract effectors[1] describe data flow abstracting over chains of components and connectors, and C&C views do not require complete interfaces with port names and types. Together, these abstraction mechanisms allow for expressive and yet intuitive specification of structural properties of C&C models. Intuitively, a C&C model satisfies a C&C view iff all elements and relations shown by the view have a satisfying concretization in the model. The formal definitions of C&C model, C&C view, and their satisfaction are available in [16] and from supporting materials [29].

Fig. 1 shows an example C&C model *Car* of a software component. It controls the car's acceleration, brake, and light signals and consists of the two subcomponents `Driving` and `ALS` (Adaptive Light System). Component `Driving` is hierarchically decomposed into three components: `ADAS` (Advanced Driver Assistance System), `ParkAssist`, and `Switch` propagating outputs of `ADAS` when driving forward and outputs of `ParkAssist` when parking. The C&C view *CA1* shown in Fig. 2 describes the `ADAS` component, which receives inputs unmodified from component `Car` (left three abstract connectors from `Car` to `ADAS`) and its `Acceleration` and `Brake` output values effect the corresponding output values of `Car`. The values of the `Brake` output port additionally effects the `Car`'s `BrakeLight` port. The C&C view *CA2* is about the `FrontLight` of the `ALS` component. It specifies that the `Acceleration` outputs of components `ADAS` and `ParkAssist` effect the front light (e.g., larger light beam as car speeds up). The model *Car* satisfies the view *CA1*.

C&C view verification, as presented in [17], gets as input a C&C model and a C&C view. Besides the Boolean answer whether the C&C model satisfies the C&C view, the verification tool produces a minimal satisfaction or one or more non-satisfaction witnesses. The *positive satisfaction witness* contains a minimal subset of the C&C model that is (1) by itself a well-formed C&C model, (2) contains all the view's components and their parent components up until their least common parent, (3) contains C&C model ports corresponding to all the view's ports, (4) contains C&C model connectors (and chains of connectors) representing all view's abstract connectors, and (5) contains C&C model data flow paths (chains of connectors and effectors) representing all view's abstract effectors. A *negative non-satisfaction witness* contains a minimal subset of the C&C model and a natural-language text, which together explain the reason for non-satisfaction. These witnesses are divided into five categories: MissingComponent, HierarchyMismatch, InterfaceMismatch, MissingConnection, MissingEffector (see [17]).

As an example, a witness for satisfaction *CA1s* is shown in Fig. 3 and demonstrates how the C&C model satisfies *CA1*.

[1] We have added the concept of effectors for our main study, see Sect. IV-A4.

Due to (2) this witness also contains the `Driving` component, due to (3) and (4) it contains the `V_Vehicle` port for the components `Car`, `Driving`, and `ADAS` as these ports belong to the connector-chain for the abstract connector going from `Car`'s `V_Vehicle` port to any port of the component `ADAS`. Due to (5) the `Switch` component is shown as it is needed in the connector-effector chain for the abstract effector going from `ADAS`'s `Acceleration` port to the corresponding `Car`'s one. The model *Car* does not satisfy the view *CA2*. A witness for non-satisfaction *CA2n* (case MissingEffector) is shown in Fig. 3. It shows all outgoing connector-effector chains starting at port `Acceleration` of component `ADAS` as well as the abstract effector's target port, `ALS`'s `FrontLight`, which is not reachable. Removing the two effectors in view *CA2* would cause the model to satisfy this modified view even though `ADAS` and `ALS` are direct siblings in the C&C view and are not direct siblings in the C&C model; C&C views allow to abstract away the intermediate component `Driving`.

III. Preliminary Study

In the preliminary study we investigate **Q1**: Which industrial contexts in automotive domain are relevant for C&C views and what challenges can the use of C&C views address? We detail this question further with subquestions: **Q1a** *What process steps in the automotive domain pose challenges that C&C views could address?*, **Q1b** *What industrial models are available for a case study?*, and **Q1c** *How and based on what artifacts can C&C views be created?*

A. Execution of Preliminary Study

To address research questions **Q1a**-**Q1c** we used the following framework. The **Objective** was to explore industrial settings in automotive domain we can use for our investigation on C&C views, and to find a relevant industry partner to participate in our main study. Specifically, we were interested in the challenges developers have to deal with during the development process involving C&C models; and we also looked for industrial data that we can use. The **Case** we studied was the development process of automotive software with the use of C&C models. The **Theory** context of our case study was the ability of C&C views to describe structural constraints on C&C models [17]. As **Method** we chose the following activities: establish contact with previous industrial partners and explain the aim of our study. After that we planned to hold a 2-3 days workshop to introduce C&C views based on examples to the industrial partner. In interviews and informal discussions we learned about the current development process of the industrial partner. Together we hypothesized what existing challenges C&C views might address. Crucial to our main study, we planned to work on real industrial data (this is an obstacle for most studies due to proprietary intellectual property concerns). Our **Selection Strategy** of industrial contacts was based on former and current research collaborations of the authors. We limited selection to collaborators working full-time for at least two years at an industrial partner. These companies included three German car manufacturers and two automotive suppliers.

For the following reasons, Daimler AG turned out to be the most promising industrial partner for our experiment. First, we had previous collaborations on evolution of Simulink models [3], [23]. Second, Daimler AG created for the federal industry-research project SPES_XT [15] industry demonstrator models. Finally, and of importance for the dissemination our work, the available models demonstrate automotive features, which are not only comprehensible by domain experts.

B. Results of Preliminary Study

In the following, we briefly summarize the results of our preliminary study. Sect. III-B1 and Sect. III-B2 addresses question *Q1a* by describing an excerpt of the development process and by listing the specific challenges we identified. In Sect. III-B3 we address question *Q1b* by describing the two Simulink models that Daimler AG agreed to release to the public.

Based on the above, we can answer *Q1c*: C&C views can be created on given Simulink models and on textual requirements of these Simulink models.

1) Industrial Development Process

We briefly describe selected development phases used at Daimler AG as they relate to our study. These phases involve three different kinds of artifacts: textual requirements, informal graphical design models, and Simulink block diagrams. We sketch the phases as follows: **Requirement Phase** First requirements of the functions to be developed are derived and documented as user-features; **Design Model Phase** Based on the requirements, the basic architecture of components and signals is derived, including high-level interaction with the environment; **Implementation Phase** Based on the Design Model, the functionality is implemented using Simulink; the individual units satisfying each requirement are developed iteratively by different software engineers; **Unit Test Phase** Software engineers derive unit tests based on the requirements they implemented; this enables them to directly test the Simulink subsystem they created; **Software Component Test Phase** Finally, after all units (e.g., `Tempomat`, `Limiter`, `FollowToStop`) have been modeled and tested, the entire Simulink model is tested against its requirements.

2) Identified Challenges

As part of our preliminary study we discussed existing challenges of C&C models development. The first listed author, identified that the following challenges have a potential to be addressed using C&C views.

III-B2a Traceability. Traceability is the ability to link between artifacts that impact each other [1]. In our context the challenge is to link a requirement with the Simulink model elements that implement it. We identified different scenarios at Daimler AG where traceability poses a challenge: First, when **preparing a technical review** of the implementation of a requirement, the engineer has to locate relevant blocks and their

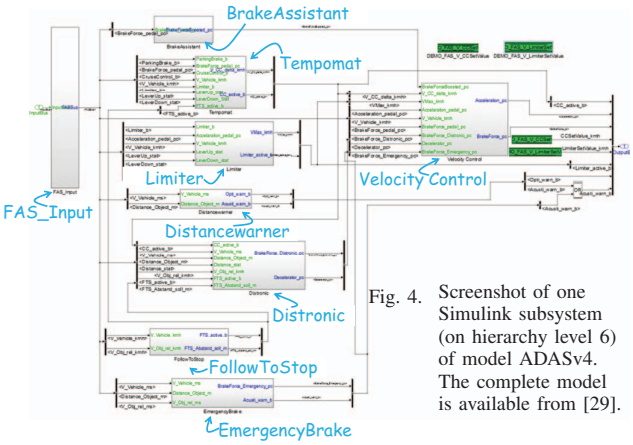

Fig. 4. Screenshot of one Simulink subsystem (on hierarchy level 6) of model ADASv4. The complete model is available from [29].

interactions; Second, when **understanding a requirement** to modify it or correct its implementation, the engineer has to identify relevant blocks and information flows to work with; Finally, when **testing the implementation of a requirement**, a software tester has to trace the requirement to the relevant subsystems and ports it has to test.

At present, engineers we interviewed at Daimler AG add tracing information (Info blocks) to Simulink subsystems. These blocks list the IDs of (and automatically link to) textual requirements implemented by the subsystem.

III-B2b Evolution. Evolution is the repeated change of software for various reasons [2]. In our context the challenge is: given a change to the model or a change to requirements, how will it impact the existing model, the requirements, and the traceability between them? We identified different scenarios at Daimler AG where evolution poses a challenge: First, when adding or **changing a requirement** the engineer needs to learn about the existing implementation and how the implementation is constrained by other requirements. Second, when **changing the model**, e.g., to implement a requirement change, the engineer wants to ensure that no other requirements are violated. Finally, models are **refactored** for various reasons, e.g., too large subsystems should be split up or names of signals change for technical reasons; engineers want to check whether the refactored model still satisfies the requirements.

According to a process description from [14], after a requirement change engineers at Daimler AG manually determine whether an implementation is compatible with the new version of the requirement. Due to the systems' complexity it is often very hard to validate this information later on.

3) Available Models

III-B3a Advanced Driver Assistance System (ADAS) This system gets as input user commands, such as brake/acceleration pedal angle, park brake activation, and movement direction (up, down, forward, backward) of cruise control lever, and sensor values such as actual vehicle speed, distance, and speed of detected objects in front of the car. Based on these inputs the ADAS calculates car's acceleration or brake force values as well as feedback to show to the driver such as warning signals and control lamp status values.

TABLE I
STATISTICS FROM ANALYZED SIMULINK MODELS (SEE SECT. III-B3) AND SIZES OF THEIR TRANSLATION TO C&C MODELS (SEE SECT. IV-A3)

Model Name	Blocks (no ports)	Port Blocks	Subsystems	Info Blocks	Depth	Requirements	Views	Components	Ports
ADASv1	327	701	122	27	12	33	17	639	1 776
ADASv2	686	1 454	211	43	12	n.a.	n.a.	2 309	9 009
ADASv3	664	1 480	203	49	12	n.a.	n.a.	2 278	8 981
ADASv4	655	1 513	195	48	13	68	26	1 396	4 438
ALS	961	2 753	184	24	10	82	7	1 086	3 193

We have four different versions of the system. **ADASv1** has the following user functions: (1) cruise control so that the car maintains the user's set speed, (2) and a limiter so that the car will not exceed a set speed. **ADASv2** extends the cruise control to a two-stage cruise control-lever and adds (3) brake assistance functionality. In **ADASv3**, cruise control gets brake support to maintain safety distance, and (4) sign detection plus (5) distance warning is added. In **ADASv4**, traffic (6) jam following, (7) distronic, and (8) emergency brake functionality is added. Fig. 4 shows a Simulink subsystem (on hierarchy level 6) of ADASv4, where these eight user functions are modeled as Simulink subsystems.

An example execution of the industrial process, as defined in Sect. III-B1, and focusing on the two above mentioned challenges, traceability and evolution, is available from [29]; it shows how the different ADAS versions could be developed at Daimler AG.

We report the sizes of the models in Tbl. I as the number of (i) Simulink blocks (including atomic blocks and subsystems, but excluding `inport` and `outport` blocks), (ii) port blocks, (iii) subsystems, and (iv) info blocks, as well as depth of the (v) subsystem hierarchy, and the number of (vi) requirements. As requirements we count distinct DOORS requirement identifiers [22] in requirements documents provided by Daimler AG; requirement documents for ADASv2 and ADASv3 were not available to us. A more detailed statistic is available from [29].

III-B3b Adaptive Light System (ALS) This model controls adaptive high and low beam, turn signals as well as cornering and ambient light [27]. Adaptive high and low beam adjust headlamps to the traffic situation and provides optimized illumination without dazzling others. Cornering light illuminates the area to the side of a vehicle to take a look around the bend. Ambient light welcomes the driver with an indirect light.

We report the size of the ALS model in the last row of Tbl. I. It has more blocks than any ADAS version and implements more requirements. However, it has less info blocks than even the smallest ADAS version. The industrial partner reports that the info blocks in ALS implement more complex functionality than in any version of ADAS.

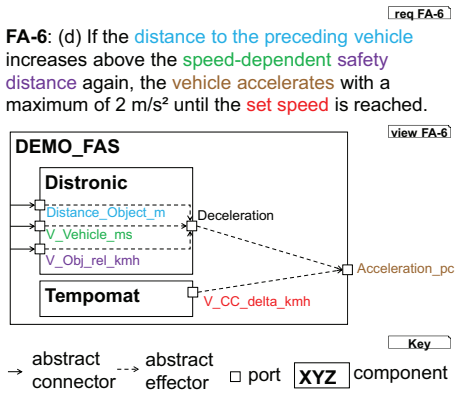

Fig. 5. Requirement **FA-6** of unit `Distronic` of ADASv4 (top) and the view created for the requirement by the domain experts (bottom)

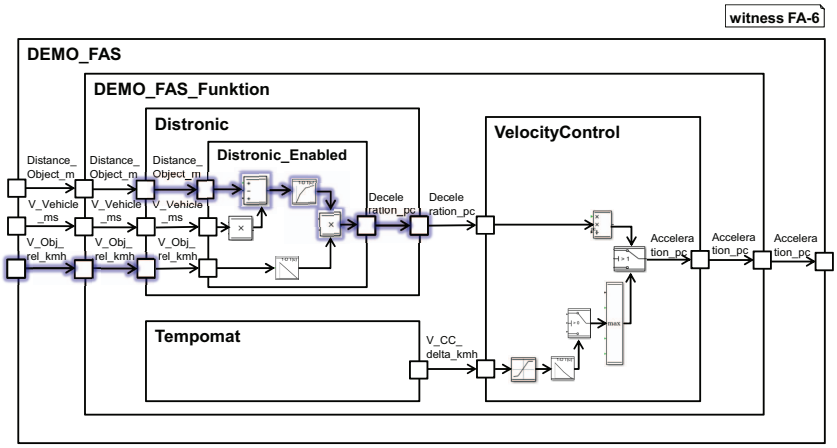

Fig. 6. Satisfaction Witness of view FA-6

IV. Main Study

The **Objective** of our main study was to evaluate the use and potential benefits of C&C views in settings similar to the development process of Daimler AG (see Sect. III-B1). The **Case** we studied was how domain experts create and use C&C views to address the challenges of traceability and evolution (see Sect. III-B2). The **Theory** context of our main study were the findings from the preliminary study and the language and tools of C&C views [16], [17]. As **Method** to collect data we recorded observations from collaboratively creating C&C views with domain experts, we presented generated witnesses to domain experts and discussed findings in interviews, we measured the time to create views, and collected statistics of verification times and sizes of generated witnesses. Our **Selection Strategy** was dominated by the availability of Simulink models and requirements documents (as described in the preliminary study above). We were not able to obtain informal design models nor traceability information within Simulink models. We have collected all C&C views created by the domain experts.

Recall the remaining three research questions of our paper:

Q2 Can domain experts create C&C views with reasonable effort and are they missing any language features?

Q3 Is C&C views verification applicable to automotive industry models and does it scale to deal with their size?

Q4 Are the verification outputs of use for the engineers?

To address **Q2** we set up an experiment with automotive domain experts to create C&C views in the context of addressing the two challenges of traceability and evolution identified in the preliminary study. Specifically, we investigated **Q2a** *How much knowledge/training of C&C views is necessary?*, **Q2b** *How well does the domain expert have to know the models to create C&C views?*, **Q2c** *How long does it take to create a C&C view?*, and **Q2d** *What missing features would domain experts like to have in C&C views (verification)?*. We report on the execution of this study in the following sections.

For each of the two identified challenges, traceability and evolution, we devised an hypothesis of how C&C views verification can assist engineers in developing and maintaining C&C models. To execute the study we had to address **Q3** and evaluate the applicability of C&C views verification on industrial models. Specifically, we investigated **Q3a** *What is the effort to use industrial Simulink models as input for C&C views verification?* and **Q3b** *Does the verification scale on industrial models?* Finally, we address **Q4** where domain experts are presented with results of C&C views verification.

We describe our experience in trying to address the above challenges, on the described models, using C&C views and related tools. All the C&C views in the study were created by the first listed author, who has been working for Daimler AG for 2 years, assisted by the last listed author, who has 3 years of experience in automotive software engineering projects. In the remainder we refer to these two authors as domain experts.

A. Main Study Execution

1) Addressing the Traceability Challenge

To address the challenge of traceability we hypothesize that given a requirement, (1) engineers can create a C&C view on the implementation of this requirement with reasonable effort, and (2) the C&C view and witnesses generated by verification assist engineers in identifying relevant Simulink blocks and signal lines (connectors) that implement the requirement.

To examine our first hypothesis we asked the domain experts to create C&C views based on textual requirements and the Simulink models of ADAS and ALS. We expected that the domain experts can create a C&C view for each requirement in less than an hour per view. Specifically, we asked the domain experts to create a C&C view *for every* ADASv1 and ADASv4 requirement. In addition, we asked the domain experts to also create C&C views for requirements of ALS, for which the first listed author focused on requirements related to under- and over-voltages. We measured the time it took domain experts to create C&C views and asked them to rate the effort.

To examine our second hypothesis, i.e., that C&C views can help trace requirements to implementations, we set up a two stage experiment. The experiment compares a C&C views independent, perfect traceability as imagined by the domain experts, to the results of C&C views verification. In the first stage, the domain experts selected 10 requirements of ADASv1 and ADASv4 at random. For each of these requirements the domain experts created a copy of the complete Simulink model and colored all elements addressed by each requirement. This step was done without referring to any C&C views. In the second stage, we executed C&C views verification on the view of the requirement and compared the generated positive witness against the "perfect traceability" coloring by the domain experts. Finally, we also asked for general observations and how their expectations were met or not met by the generated witnesses of C&C views verification.

IV-A1a Example of a requirement, C&C view, and witness Fig. 5 shows requirement **FA-6** of ADASv4 and its C&C view, which describes a feature of units `Distronic` and `Tempomat`. The C&C view shows these two units contained in software component (SWC) `DEMO_FAS`[2], the main SWC of ADASv4. The view also shows that `Tempomat`'s three inputs `Distance_Object_m`, `V_Vehicle_ms`, and `V_Obj_rel_kmh` originate directly (without being modified) from the main SWC `DEMO_FAS` and influence the `Deceleration` value. These connections are not direct in the C&C model. The effectors from `Distronic`'s `Deceleration` and from `Tempomat`'s `V_CC_delta_kmh` to `DEMO_FAS`'s `Acceleration_pc` show that these values influence the car's acceleration. The `Deceleration` value is not part of the requirement. However, the domain experts decided to include it in the C&C view because deceleration is a limiting factor of the car's acceleration and thus an important design concept to understand the implementation of the requirement.

Fig. 6 shows the generated witness for ADASv4 and the C&C view shown in Fig. 5. The abstract connector going from `DEMO_FAS` (unknown port) to `Distronic`'s `V_Obj_rel_kmh` port in the view is the bottom highlighted connector chain in the witness. The abstract effector starting at `Distronic`'s `Distance_Object_m` and ending at `Distronic`'s `Deceleration_pc` in the view is satisfied by the upper highlighted connector-component chain in Fig. 6. This generated witness shows that all components, ports, and abstract connectors, as well as abstract effectors in the view, are also present in the model. The witness shows for each abstract connector and effector a shortest path in the model.

Note that our verification tool [17] reads as input textual C&C views and produces textual witness [10]. We transcribed graphical views created by the domain experts to their textual representation, and selected witnesses generated by the tool were manually translated back to a graphical representation for inspection by the domain expert working at Daimler AG.

IV-A1b Design decisions for creating C&C views If an output port of a component unit has been mentioned in the requirement text, the domain experts added the unit which receives this output to the view; in this way the view emphasizes high-level component interaction showing user function dependencies. They followed the same principle if an input signal was mentioned and added the component sending the input to the view. Most textual requirements have a trigger-action pattern (*if* sentences); consequently, the created views contain an effector from the port whose values are the trigger to the port whose actions are a response to the trigger.

[2]FAS is an abbreviation of the single German word for ADAS.

TABLE II
SIZES OF C&C VIEWS CREATED BY DOMAIN EXPERTS AND WITNESSES GENERATED BY C&C VIEWS VERIFICATION (COMPLETE DATA AT [29])

(a) size distribution of 17/26 views

#Elements	Components(%)		Connectors (%)		Effectors (%)		Ports (%)	
	v1	v4	v1	v4	v1	v4	v1	v4
0	0	0	0	4	12	4	6	0
1	0	0	0	4	76	54	0	4
2	24	27	41	35	6	15	29	19
3	71	54	41	35	6	4	41	42
4	6	15	12	8	0	4	12	8
5	0	0	6	8	0	4	12	15
6	0	0	0	0	0	12	0	4
7	0	0	0	0	0	0	0	4
8	0	0	0	4	0	0	0	0
9	0	4	0	4	0	0	0	4
10	0	0	0	0	0	4	0	0

(b) size distribution of 17/26 witnesses

#Elements	Components(%)		Connectors (%)		Effectors (%)		Ports (%)	
	v1	v4	v1	v4	v1	v4	v1	v4
0-5	0	0	6	4	35	4	0	0
6-11	18	0	0	4	53	54	6	4
12-17	29	27	12	38	12	15	0	19
18-23	29	58	18	31	0	4	12	42
24-29	24	12	18	8	0	4	18	8
30-35	0	0	35	8	0	0	6	15
36-41	0	0	6	0	0	15	18	4
42-47	0	0	6	0	0	0	18	4
48-53	0	0	0	4	0	0	18	0
54-59	0	4	0	4	0	0	6	4
≥60	0	0	0	0	0	4	0	0

Percentage of **(a)** C&C views and **(b)** witnesses with y elements (column *#Elements*) of kind x (caption of column), e.g., 71% of C&C views of ADASv1 have 3 components.

In total, the domain experts have created 17 C&C views for ADASv1, 26 C&C views for ADASv4, and 7 C&C views for ALS (recall that the requirements for ADASv2 and ADASv3 were not available). Tbl. II (a) gives an overview of the sizes of the C&C views created by the domain experts. Specifically, the table shows the distribution of numbers of C&C views elements (components, connectors, effectors, and ports) in all ADAS views. It shows that most views, have three components[3], two or three connectors, one effector, and three ports (see highlighted cells in Tbl. II (a)). Although the architecture of ADASv4 is more complex than the architecture of ADASv1, only few views are larger (e.g., there exist one view with nine components and another with ten effectors).

2) Addressing the Evolution Challenge

To address the challenge of evolution we hypothesize that (1) C&C views verification is useful to detect the violation of a requirement and that (2) the generated witness can help the engineer to locate and understand the reasons for the violation in the C&C model. Specifically, given one version of a C&C model that satisfies all C&C views, and another, updated version of the C&C model, we hypothesize that a negative verification result points to the possible violation of a requirement or the design decisions of its implementation.

To examine our hypotheses we checked whether the C&C models evolved from ADASv1 (ADASv2 to ADASv4) still satisfy the C&C views of requirements of ADASv1 (we expected positive verification results). In addition, we checked whether versions prior to ADASv4 satisfy the C&C views of requirements of ADASv4 (we expected negative results

[3]71% of all views of ADASv1 and 54% of all views of ADASv4 contain three components

for requirements not implemented in versions ADASv1 to ADASv3 and positive results for implemented requirements).

First, the domain experts reviewed changes in the features and requirements and identified which C&C views of ADASv1 should be satisfied by ADASv2. They expected that 5 C&C views related to the cruise control lever, which was documented to have changed in ADASv2 (see Sect. III-B3a), would not be satisfied by ADASv2. Then we verified ADASv2 against 17 C&C views for requirements of ADASv1. As a result, verification failed for 12 views including the expected 5 views mentioning the cruise control lever. The view `FA-19`, among others expected to be satisfied, failed verification with message *No match for port* `CC_active_b` *of component* `Tempomat`. We found out that the signal names `CC_active_b` and `Limiter_active_b` had changed to `CC_enabled_b` and `Limiter_enabled_b` in ADASv2. After renaming these signals in ADASv2 all C&C views (including the 5 that should not) produced positive verification results. After further investigation, the domain experts discovered that contrary to the documented requirements, the change of the cruise control lever was only implemented in ADASv3 and ADASv4, and only there the respective 5 views failed verification.

As a second step, we asked domain experts to review the features added from ADASv3 to ADASv4 and identify which C&C views of ADASv4 should not be satisfied by ADASv3 and why. Then we verified all C&C views of ADASv4 against ADASv3. Our verification revealed again a mismatch of the names of signals `CC_active_b` and `Limiter_active_b`. It turns out that these signals have the same names in ADASv1 and ADASv4 but different names in ADASv2 and ADASv3. After we updated the signal names, verification failed exactly on 5 C&C views that describe the emergency brake and the follow to stop features added only in ADASv4, as expected.

3) Translating Simulink Block Diagrams to C&C Models

All models provided by Daimler AG for our study are Simulink block diagrams. Although Simulink block diagrams appear to be very similar to our C&C models, where blocks and subsystems are components, ports are ports, and signal lines are connectors, the relation turned out to be more complicated. Simulink uses blocks not only to model components but also for many other concerns, e.g., to model variability and conditional execution. In addition, connectors are not the only way of interaction, as Simulink has `Data Store` blocks acting as global variables. Our evaluation with the C&C views verification prototype [17] requires a translation of Simulink block diagrams to C&C models. We now describe the main technical challenges of the translation (see [6] for implementation details) and how our automated translation addressed them.

IV-A3a Model references. Simulink block diagrams may contain `Model blocks`, which reference blocks from libraries. This referencing mechanism does not exist in C&C models. Thus, in a first translation step we replace each Simulink model block by a copy of the block it references (Simulink uses the same mechanism for simulation and code generation).

IV-A3b Namespace. In contrast to C&C models, having a flat namespace, Simulink has a hierarchical one (e.g.,the block name `DEMO_FAS` appears three times in all versions of ADAS). To avoid encoding component hierarchy in component names, we decided not to use Simulink's full-qualified names, and thus the translation appends a running number to block names appearing multiple times in the Simulink model.

IV-A3c Simulink specific blocks. Simulink supports special blocks that do not directly correspond to components in a C&C model. Examples include `Data read`, and `Data write` blocks allowing interaction between different subsystems without using connectors. Furthermore, models provided by Daimler AG use conditional execution (`If block`) and reconfiguration (`Enabled subsystems`) blocks for product-line modeling. Thus, first all special blocks are transformed to behavior equivalent subsystems including only standard blocks and connectors. Second, these are translated to C&C models.

IV-A3d Block names and shown names. Simulink is a visual modeling language where displayed text and name of elements do not always agree. Port blocks may even have the same display name, in which case Simulink automatically makes port names unique by appending the port's direction and a number to its name. Our translation addresses this issue by using displayed names of elements, as long as they do not contradict well-formedness rules of C&C models.

IV-A3e Signal buses. For graphical overview purposes engineers use (even nested) signal buses to group signals going from one subsystem to another. Since these buses are motivated to not clutter the graphical representation, we remove all `Bus Creator` and `Bus Selector` blocks and connect the subsystems' output and input ports directly to make connections between components explicit.

IV-A3f Translation results. The last column of Tbl. I includes the sizes of C&C models resulting from our automated translation. The translation increases the size of all models as measured in number of blocks and ports compared to number of components and ports. The increase depends on the specific types of Simulink blocks used in the models. The combined increase ranges from factor 1.2 for ALS to 5.3 for ADASv2. Complete statistics with breakdown on types of Simulink blocks are available from [29].

4) Extending C&C Views Verification

The technical challenges to obtain C&C models from Simulink models and important requirements from Daimler have led us to extend the C&C views verification prototype tool implementation from [17] in two ways.

IV-A4a Names from Simulink. As a necessity, the domain experts need to create C&C views with names they know from the Simulink model (although these might be displayed names and not element names). Most cases for enabling the use of displayed names are handled by our translation. The only remaining cases for ADAS and ALS models were component names with appended running numbers. The matching of

TABLE III
VERIFICATION AND WITNESS GENERATION TIMES REPORTED AS AVERAGES OVER ALL C&C VIEWS IN MS

Model	pos. Verification	neg. Verification	one sat. Witness	all non-sat. Witnesses
ADASv1	62 ms	61 ms	27 ms	42 ms
ADASv2	1 809 ms	1 174 ms	615 ms	6 443 ms
ADASv3	1 459 ms	1 303 ms	566 ms	5 928 ms
ADASv4	404 ms	506 ms	114 ms	963 ms
ALS	218 ms	126 ms	82 ms	175 ms

displayed names to components is not unique. We handle these cases by analyzing C&C views before verification. For every component in a C&C view we compute all possible matching component names in the C&C model. We then create one view for every combination of matching names and execute C&C views verification. Verification is successful iff the model satisfies the view for at least one combination of matches. This existential interpretation is a sound extension of [17].

As an example, ADASv1 has three blocks with name `DEMO_FAS` (translated to C&C model components `DEMO_FAS_1` to `DEMO_FAS_3`). A C&C view that connects `VelocityControl` to `DEMO_FAS` results in three views we check. Connecting `DEMO_FAS` to `DEMO_FAS` in a C&C view results in nine combinations. Our prototype reports the smallest witness for satisfaction of all satisfied combinations.

IV-A4b Effectors. C&C views as presented in [17] did not feature the concept of effectors. Indeed, after initial experiments of creating C&C views, the domain experts identified a necessity to express interaction between components and ports that goes beyond chains of connectors. Abstract connectors of C&C views express a directed chain of connectors, i.e., only the transportation of information. In contrast, the domain experts wanted to express effects that can include computations in components. We have thus added the concept of effectors to C&C views definition and verification. Formal definitions are available from [29]. As an approximation of unknown computations in atomic components of the C&C model, our translation adds effectors between all input ports and output ports (other components are not modified as their computations are defined by composition).

V. RESULTS OF MAIN STUDY AND LESSONS LEARNED

We now present results to answer research questions *Q2* to *Q4*, results from addressing the challenges of traceability and evolution, additional observations, and threats to validity.

A. *Q2* Feasibility and Effort to Create C&C Views

First, we observed that **many but not all requirements allow to capture design decisions of their implementation by C&C views**. Specifically, UI-related and extra-functional requirements, e.g., *"FA-53: The safety classification of the system speed control is ASIL B."*, are not covered. The domain experts created 17 C&C views for 21 out of 33 requirements of ADASv1 and 26 C&C views for 50 out of 68 requirements of ADASv4.

For some requirements, e.g., FA-67, FA-68, and FA-89, the domain experts chose to create a single, common view. Overall, the domain experts added supporting C&C views for 70% (71 out of 101) of the requirements of ADASv1 and ADASv4. Interestingly, for the more complex requirements of ALS the domain experts suggested to create multiple C&C views for a single requirement.

Second, we conclude that **domain experts can create C&C views with reasonable effort**: On average the domain experts needed 30 minutes to create a C&C view for a given requirement. Because most requirements address high-level Simulink subsystems and C&C views only represent structural properties, not much time was spent on investigating low-level behavior implementations.

B. *Q3* Technical Applicability

We had perceived Simulink models to be very similar to C&C models. We significantly underestimated the technical efforts[4] required to transform Simulink models to C&C models. However, we were finally able to translate all Simulink models to equivalent C&C models (see Sect. V-F for validation).

To answer the research questions: **C&C views verification can now be applied to Simulink models**. A few challenges remain to go from graphical formats used by engineers to textual formats used by our prototype tools. As an example, generated witnesses are witnesses on the translated C&C model, where some Simulink blocks have been replaced by components with no direct equivalent in Simulink. These components made the translation of witness as tracing information for Simulink challenging.

Tbl. III shows the average times for C&C views verification, split into positive and negative results. It also reports on the time needed to generate one witness for satisfaction or possibly multiple witnesses for non-satisfaction [17]. **These results show that C&C views verification is fast.** The most time consuming task is the generation of witnesses for non-satisfaction (up to on average 6s for ADASv2).[5] The verification itself takes on average less than 2s and the generation of witnesses for satisfaction is very fast (below 1s). The average times for ADASv2 and ADASv3 are greater than for ADASv1 and ADASv4 as expected from their different sizes, as shown in Tbl. I (individual times available from [29]).

In our preliminary study, the domain experts requested the addition of effectors (see Sect. IV-A4b). This concept was so important that we added it for the main study. As shown in Tbl. II, **the domain experts used abstract effectors in almost all C&C views** for documenting interaction of subsystems.

C. *Q4* Helpfulness of Witnesses

For our evaluation of helpfulness of witnesses of satisfaction and non-satisfaction generated by C&C views verification we have manually translated the textual output of our tool to graphical C&C views for discussion with the domain experts.

Tbl. II (b) shows the sizes of generated witnesses for satisfaction (for positive verification results). It is interesting to observe that on average the witnesses (see highlighted

[4]The implementation of our translation tool required one person year.
[5]We did not yet implement witness generation for missing effectors.

cells in Tbl. II (b)) for ADASv1 have more connectors and effectors than the witnesses for ADASv4, i.e., **larger C&C views and a more complex model do not necessarily lead to larger witnesses**. Nevertheless, the numbers show that witnesses for satisfaction are much larger than C&C views created by the domain experts (note the different scales in Tbl. II (a) from 0 to 10 and Tbl. II (b) from 0 to $\geq$ 60). Surprisingly, the large amount of component, connector, and port elements in the graphical witness were – in contrast to our first beliefs – no problem for the domain experts. They are used to large graphical models from Simulink, e.g., the one shown in Fig. III-B3a. **The domain experts found the witnesses for satisfaction very helpful (see also Sect. V-D1).**

To address the challenge of evolution we also presented negative verification results to the domain experts, including generated non-satisfaction witnesses with their natural-language descriptions. We observed that **the domain experts mainly relied on our tool-generated natural-language descriptions**. After reading the natural-language descriptions, they directly opened the Simulink model. We conclude that **the natural-language descriptions that we generate with the witnesses for non-satisfaction are useful.**

D. Results from Addressing the Identified Challenges

1) Traceability

In our study we found that **views can successfully be applied to address the challenge of traceability**.

Specifically, we found out that verification helps to uncover defects during the creation of C&C views for traceability. The domain experts found typos in signal names in the Simulink models and discovered an inconsistent type encoded in a signal name. **The domain experts also found the generated witnesses for satisfaction of C&C views helpful to trace a requirement to its implementation**.

However, and maybe biased by the setup of our experiment (see Sect. IV-A1), the domain experts expected or **wished the generated witness would be a complete excerpt of the implementation** of a requirement. However, witnesses generated by C&C views verification demonstrate satisfaction of an abstract effector by showing exactly one (and not all) satisfying chains of connectors and involved components. As an example, the domain experts found a missing Simulink subsystem in `Limiter_SetValue` when investigating effector `LeverDown_stat -> VMax_kmh` of FA-67. The part showing the missing component was not included in the generated witness. A more complete and different form of traceability could go beyond structure and include analyses of behavior.

Compared to existing traceability features for Simulink models, **the domain experts positively valued abstract connectors and effectors**. Existing traceability information is limited to blocks and does neither support tracing connectors nor their more convenient abstract counterparts in C&C views. We conclude that traceability for interaction is a valuable extension of traceability information provided by C&C views.

2) Evolution

C&C views successfully addressed evolution related challenges, confirming our two hypotheses. In the two experiments we conducted, C&C views verification was applied successfully to (i) ensure that during model evolution unchanged requirements were not violated (C&C verification succeeds) and to (ii) check whether an evolved requirement has been implemented (by testing whether previously valid and now invalid requirements indeed fail due to evolution). Specifically, as reported in Sect. IV-A2, **we found that one requirement was not implemented** (in contrast to the changelog), and we found inconsistencies of signal names within versions. **C&C views verification validated the design of all implemented requirements.**

It is important to note that C&C views verification only checks what is specified in the C&C views. It does not verify behavioral properties of an implementation and is thus not addressing all challenges of evolution [20]. Nevertheless, our experiment showed that based on the reuse of all C&C views domain experts had **no additional overhead** for addressing evolution; making the view verification in this combination more interesting for industry.

E. Additional Observations and Desired Extensions

In interviews we learned that engineers at Daimler AG create Simulink models with manually highlighted blocks or manually deleted elements in order to show only important information (slices), e.g., for discussing implementations of requirements or to locate defects. Defect slices narrow the focus on elements causing an error and might contain many details of subcomponents and atomic blocks. We believe that **C&C views can also be used to highlight important blocks involved in defects** and automatically generate defect slices. However, we did not evaluate potential benefits in our study.

Our industrial partner also wished to tag abstract effectors with conditions such as `v >= 20 km/h`, in order to match only component-connector chains for features that are enabled for speeds above 20 km/h (e.g., distance control). A witness would identify subsystems active at high speeds.

Finally, **we came across industrial modeling features an engineer might want to express in C&C views**. These features include different modes of subsystems, e.g., over- or undervoltage mode in ALS, and dynamic product lines, e.g., disabled and enabled subsystems of ADASv4.

F. Threats to Validity

Since the ADAS and ALS models were released by Daimler AG for demonstration and evaluation purposes, they might have been modified in complexity and functionality from the original models. We have no means to address this threat for the internal validity of our study.

Since Daimler AG is a large company with over 280,000 employees, and we only gained insight into one specific software development process, other departments might use processes and methods different from the ones we describe in this paper. This limits the generalizability of our findings.

We were not able to evaluate the use of C&C views on pure C&C models, but only on Simulink block diagrams, a specific variant of C&C models with many more technical details. To mitigate this threat, we describe in Sect. IV-A3 how we translated Simulink block diagrams to C&C models. This translation is a necessary prerequisite for applying C&C views verification in industry. The domain experts used Simulink models and did not report the technical difference as an issue for creating views and understanding verification results.

Still, from Tbl. I it is clear that our translation from Simulink to C&C models significantly changes the sizes of the models. This likely has an impact on the sizes of witnesses shown in Tbl. II and verification times reported in Tbl. III. To reduce this impact we have carefully developed translations for each Simulink feature, to yield a correct and generic translation that minimizes component creation. We have validated the behavioral equivalence of all translation results by executing more than thousand automated back-to-back tests comparing the outputs of code generated from Simulink and code generated from our C&C models for thousands of inputs. These helped us in mitigating this threat.

It is important to note that our study worked on existing models. Yet, our hypotheses for addressing the challenges of traceability and evolution are based on using C&C views during the development process and not after the fact on existing models. Regarding evolution, only major revisions of models were available to us while in an ordinary development process, changes are likely smaller and incremental.

Finally, we have conducted all experiments in a very specific context, i.e., control systems from the automotive domain modeled in Simulink, and with only two domain experts. To mitigate this threat we were able to include two different systems in our study, the Adaptive Light System and Advanced Driving Assistance System. We would like to emphasize that this scope and making all materials available goes beyond many comparable studies involving real industrial models.

VI. Related Work

Two developers at Daimler AG stated the following challenges in an insight report [26]: (a) presentation of requirements is a problem, (b) daily requirement discussions result in possible changes and manually updates of trace links, and (c) the coupling between document- and model-based tools lacks automation. These challenges are similar to the ones we identified (see Sect. III-B2). C&C views can help to address these problems by providing (a) graphical representation of structure and interaction, (b) trace-links in generated witnesses, and (c) automated C&C views verification.

A similar setup to our case study for tracing textual requirements to models was reported by Briand et. al. [5]. Textual requirements are decomposed and enriched with trace links. In a controlled experiment, they showed that slices generated by their SafeSlice tool [7] support safety inspections. We found similar benefits for inspection and the discovery of defects in our setting, where trace links can be seen as C&C views and generated slices as witnesses from C&C views verification.

There exist several studies dealing with requirements/use cases evaluated on automotive models such as behavioral verification at Electronic Brake Management by BMW AG [4], analyzing user intentions in in-car infotainment system by VW AG [13], introducing use case product line modeling in automotive sensor systems (e.g. Driver Presence Detection) by IEE SA [11]. Similar to our study the setting of these works is the automotive domain. However, the evaluated models and techniques are very different.

Recently, we presented C&C views [16], as a means to formally and intuitively specify constraints on the structure of C&C models. The verification problem of checking a C&C model against a view was investigated in [17]. The synthesis problem of automatically generating a C&C model satisfying a given C&C views specification, if one exists, was studied in [16]. Both works on verification and synthesis were evaluated only on synthetically generated or smaller C&C models. Their application had neither been studied in an industrial context nor evaluated with an industrial partner.

VII. Conclusion

C&C views with verification and synthesis techniques have been recently suggested as a means for formal yet intuitive structural specification of C&C models. In this paper we present our experience in applying C&C view verification in an industrial automotive setting at Daimler AG. Even though our case study focused on the automotive domain, most of our findings on C&C views verification could also apply for other industrial domains dealing with distributed control in embedded software or cyber-physical systems, e.g., robotics, production systems, or telecommunication.

We discovered promising applications of C&C view verification for supporting requirements tracing and evolution. In our experiments, domain experts created graphical C&C views from textual requirements with reasonable effort.

Our case study revealed and addressed technical challenges in translating Simulink models to C&C models. The sizes of translated, industrial C&C models were no problem for C&C views verification with average verification times below 2s. Domain experts found the generated descriptions of verification results and the graphic representations of generated witnesses very helpful and discovered inconsistencies between requirements and their implementation. Our concerns about the large sizes of witnesses turned out to be unnecessary.

As part of the study, we extended C&C views and its verification with abstract effectors to model data flow between components. This feature was requested by our industrial partner and appeared in almost every C&C view created.

Finally, we discovered additional use cases, e.g., for modeling error slices with C&C views. Our industrial partner also suggested extensions that include combinations with behavior analyses and rich specification mechanisms for modes and reconfiguration in C&C views.

Acknowledgments This research was supported by a Grant from the GIF, the German-Israeli Foundation for Scientific Research and Development.

REFERENCES

[1] M. F. Bashir and M. A. Qadir. Traceability Techniques: A Critical Study. In *IEEE International Multitopic Conference*, pages 265–268, Dec 2006.

[2] K. H. Bennett and V. Rajlich. Software maintenance and evolution: a roadmap. In A. Finkelstein, editor, *22nd International Conference on on Software Engineering, Future of Software Engineering Track, ICSE 2000, Limerick Ireland, June 4-11, 2000.*, pages 73–87. ACM, 2000.

[3] V. Bertram, P. Manhart, D. Plotnikov, B. Rumpe, C. Schulze, and M. von Wenckstern. Infrastructure to use OCL for runtime structural compatibility checks of simulink models. In A. Oberweis and R. H. Reussner, editors, *Modellierung 2016, 2.-4. März 2016, Karlsruhe*, volume 254 of *LNI*, pages 109–116. GI, 2016.

[4] T. Bienmüller, J. Bohn, H. Brinkmann, U. Brockmeyer, W. Damm, H. Hungar, and P. Jansen. Verification of automotive control units. In *Correct System Design, Recent Insight and Advances, (to Hans Langmaack on the Occasion of His Retirement from His Professorship at the University of Kiel)*, pages 319–341, London, UK, UK, 1999. Springer-Verlag.

[5] L. Briand, D. Falessi, S. Nejati, M. Sabetzadeh, and T. Yue. Traceability and sysml design slices to support safety inspections: A controlled experiment. *ACM Trans. Softw. Eng. Methodol.*, 23(1):9:1–9:43, Feb. 2014.

[6] S. Brunecker. Transforming Simulink Models to MontiArc Models. Bachelor's thesis, 2016. Available from our supporting materials website [29].

[7] D. Falessi, S. Nejati, M. Sabetzadeh, L. C. Briand, and A. Messina. Safeslice: a model slicing and design safety inspection tool for sysml. In T. Gyimóthy and A. Zeller, editors, *SIGSOFT/FSE'11 19th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-19) and ESEC'11: 13th European Software Engineering Conference (ESEC-13), Szeged, Hungary, September 5-9, 2011*, pages 460–463. ACM, 2011.

[8] P. H. Feiler and D. P. Gluch. *Model-Based Engineering with AADL - An Introduction to the SAE Architecture Analysis and Design Language*. SEI series in software engineering. Addison-Wesley, 2012.

[9] P. H. Feiler, D. P. Gluch, and J. J. Hudak. The architecture analysis & design language (AADL): An introduction. Technical report, Software Engineering Institute, Carnegie Mellon University, 2006.

[10] A. Haber, J. O. Ringert, and B. Rumpe. MontiArc - Architectural Modeling of Interactive Distributed and Cyber-Physical Systems. Technical Report AIB-2012-03, RWTH Aachen, February 2012.

[11] I. Hajri, A. Goknil, L. C. Briand, and T. Stephany. Applying product line use case modeling in an industrial automotive embedded system: Lessons learned and a refined approach. In Lethbridge et al. [12], pages 338–347.

[12] T. Lethbridge, J. Cabot, and A. Egyed, editors. *18th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MoDELS 2015, Ottawa, ON, Canada, September 30 - October 2, 2015*. IEEE Computer Society, 2015.

[13] D. Lüddecke, C. Seidl, J. Schneider, and I. Schaefer. Modeling user intentions for in-car infotainment systems using bayesian networks. In Lethbridge et al. [12], pages 378–385.

[14] P. Manhart. Systematische rekonfiguration eingebetteter softwarebasierter fahrzeugsysteme auf grundlage formalisierbarer kompatibilitätsdokumentation und merkmalbasierter komponentenmodellierung. In S. Kowalewski and B. Rumpe, editors, *Software Engineering 2013: Fachtagung des GI-Fachbereichs Softwaretechnik, 26. Februar - 2. März 2013 in Aachen*, volume 213 of *LNI*, pages 301–317. GI, 2013.

[15] P. Manhart. Schlussbericht SPES_XT : Software Plattform Embedded Systems 2020. Technical Report DOI:10.2314/GBV:87150491X, Daimler AG, 2015.

[16] S. Maoz, J. O. Ringert, and B. Rumpe. Synthesis of component and connector models from crosscutting structural views. In B. Meyer, L. Baresi, and M. Mezini, editors, *Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE'13, Saint Petersburg, Russian Federation, August 18-26, 2013*, pages 444–454. ACM, 2013.

[17] S. Maoz, J. O. Ringert, and B. Rumpe. Verifying component and connector models against crosscutting structural views. In P. Jalote, L. C. Briand, and A. van der Hoek, editors, *36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31 - June 07, 2014*, pages 95–105. ACM, 2014.

[18] Mathworks. Simulink User's Guide. Technical Report R2016b, MATLAB & SIMULINK, 2016.

[19] Mathworks. Stateflow User's Guide. Technical Report R2016b, MATLAB & SIMULINK, 2016.

[20] T. Mens, J. Magee, and B. Rumpe. Evolving software architecture descriptions of critical systems. *IEEE Computer*, 43(5):42–48, 2010.

[21] A. Pretschner, M. Broy, I. H. Krüger, and T. Stauner. Software engineering for automotive systems: A roadmap. In L. C. Briand and A. L. Wolf, editors, *International Conference on Software Engineering, ISCE 2007, Workshop on the Future of Software Engineering, FOSE 2007, May 23-25, 2007, Minneapolis, MN, USA*, pages 55–71. IEEE Computer Society, 2007.

[22] PROMETO. ISO 26262 compliant usages of IBM Rational DOORS in safety critical E/E-projects within the automotive domain. Whitepaper, IBM and PROMETO.

[23] B. Rumpe, C. Schulze, M. von Wenckstern, J. O. Ringert, and P. Manhart. Behavioral compatibility of simulink models for product line maintenance and evolution. In D. C. Schmidt, editor, *Proceedings of the 19th International Conference on Software Product Line, SPLC 2015, Nashville, TN, USA, July 20-24, 2015*, pages 141–150. ACM, 2015.

[24] P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, 2009.

[25] R. N. Taylor, N. Medvidovic, and E. Dashofy. *Software Architecture: Foundations, Theory, and Practice*. Wiley, 2009.

[26] M. Weber and J. Weisbrod. Requirements engineering in automotive development: Experiences and challenges. *IEEE Software*, 20(1):16–24, 2003.

[27] Mercedes-Benz TechCenter website. http://techcenter.mercedes-benz.com/en/. Accessed 4/2017.

[28] OMG SysML website. http://www.omgsysml.org/. Accessed 4/2017.

[29] Supporting materials for our case study. Available from http://www.se-rwth.de/materials/cncviewscasestudy/.