Software & Systems Modeling (2019) 18:3421–3459
https://doi.org/10.1007/s10270-019-00723-2

REGULAR PAPER

Improving manual reviews in function-centered engineering of embedded systems using a dedicated review model

Marian Daun[1] · **Thorsten Weyer**[1] · **Klaus Pohl**[1]

Received: 24 August 2017 / Revised: 3 December 2018 / Accepted: 24 January 2019 / Published online: 2 February 2019
© Springer-Verlag GmbH Germany, part of Springer Nature 2019

Abstract
In model-based engineering of embedded systems, manual validation activities such as reviews and inspections are needed to ensure that the system under development satisfies the stakeholder intentions. During the engineering process, changes in the stakeholder intentions typically trigger revisions of already developed and documented engineering artifacts including requirements and design specifications. In practice, changes in stakeholder intentions are often not immediately perceived and not properly documented. Moreover, they are quite often not consistently incorporated into all relevant engineering artifacts. In industry, typically manual reviews are executed to ensure that the relevant stakeholder intentions are adequately considered in the engineering artifacts. In this article, we introduce a dedicated review model to aid the reviewer in conducting manual reviews of behavioral requirements and functional design specification—two core artifacts in function-centered engineering of embedded software. To investigate whether the proposed solution is beneficial we conducted controlled experiments showing that the use of the dedicated review model can significantly increase the effectiveness and efficiency of manual reviews. Additionally, the use of the dedicated review model leads to significantly more confident decisions of the reviewers and is perceived by the reviewers as significantly more supportive compared with reviews without the dedicated review model.

Keywords Behavioral requirements · Embedded software · Functional design · Review model · Requirements engineering · Perspective-based review · Model transformations

1 Introduction

Model-based software engineering is commonly seen as an adequate means to cope with the complexity of embedded and cyber-physical systems [1–3]. The use of formal models allows automating verification activities and thereby contributes to the correctness of such systems. However, to validate whether stakeholder intentions are correctly considered in the created engineering artifacts, including requirements and design, manual reviews and inspections of the artifacts are still required. Consequently, safety standards, such as [4] or [5], mandate the use of manual validation activities to ensure that "software, documentation, or other items meet user needs and expectations, whether specified or not" [6].

Function-centered engineering is a commonly used paradigm to cope with the increasing number and complexity of embedded systems' software functions and their interdependencies (cf. [7]). In this paper, we focus on the validation of two major engineering artifacts in the development of embedded systems: behavioral requirements and functional design [8]. Both artifacts can become inconsistent in the course of the engineering process, due to changes in stakeholder intentions, which are often not sufficiently documented [9]. Therefore, model-based automated approaches are not able to detect whether behavioral requirements or the functional design correctly reflects the current stakeholders' intentions. To this end, we propose the use of a dedicated review model to support the manual validation of behavioral requirements and functional design.

In this paper we introduce a dedicated review model to aid the reviewer in conducting manual reviews of behavioral

Communicated by Dr Sebastien Gerard.

✉ Thorsten Weyer
thorsten.weyer@paluno.uni-due.de

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

Klaus Pohl
klaus.pohl@paluno.uni-due.de

[1] paluno – The Ruhr Institute for Software Technology, University of Duisburg Essen, 45127 Essen, Germany

requirements and functional design specification against the stakeholder intentions. In addition, we report on our experimental investigation on whether the use of a dedicated review model can improve the manual validation of those artifacts. The dedicated review model integrates the interaction-based behavior specified in the behavioral requirements and in the functional design into a single model. The review model highlights inconsistencies between behavioral requirements and the functional design, which are in turn indicators for behavioral properties that are invalid with respect to the stakeholder intentions.

The results obtained from three controlled experiments indicate that the use of a dedicated review model increases the review's effectiveness and efficiency as well as the reviewer's confidence in decision making. The results also indicate that the subjectively perceived supportiveness (in terms of perceived ease of use, perceived usefulness and computer self-efficacy) of a review using the review model is significantly higher than that of a review of the original specifications.

The remainder of the paper is structured as follows: In Sect. 2 we provide background information about stakeholders and the use of behavioral requirements and functional design during the engineering of software-intensive embedded systems. Section 3 elaborates on the challenges faced when validating behavioral requirements and functional design and discusses potential solutions. Section 4 introduces the dedicated review model to support the manual validation of behavioral requirements and functional design, and sketches the automated generation of the dedicated review model. Section 5 outlines the empirical evaluation setup, which defines the research questions to be validated by the experiments and outlines the design of the experiments. Section 6 presents the results of the controlled experiments conducted, and Sect. 7 reports on the principal findings obtained from the experiments. Finally, Sect. 8 discusses the related work and Sect. 9 concludes the paper.

2 Foundations

In the following, we introduce the terms "stakeholder intentions" (Sect. 2.1), "behavioral requirements" (Sect. 2.2) and "functional design" (Sect. 2.3) and provide additional background information. Thereby, we also introduce a concrete model-based representation of behavioral requirements and functional design.

2.1 Stakeholder intentions

In the sense of [6] a *stakeholder* is anybody who has a legitimate interest in the system to be developed (i.e., "*individual, team, organization or classes thereof, having an interest in a system*" [10]). This includes explicitly "end users, end user organizations, supporters, developers, producers, trainers, maintainers, disposers, acquirers, supplier organizations and regulatory bodies, interested parties, decision-makers" [6]. Hence, stakeholders are not only intended end users and involved developers but also, among others, the legislator or standard organizations.

As *stakeholder intentions* we refer to the "*needs and expectations*" of these stakeholders [11]. Needs and expectations in this sense not only refer to the end users wishes or the developers' expert knowledge but also to laws to be adhered to or other regulation to be abided by. In such cases, legislators or other authorities are stakeholders pursuing certain intentions. In particular, in the development of embedded systems, regulations such as laws or normative standards issued by legislators or other authorities frequently constitute a source for requirements [9].

Stakeholder intentions are frequently contradictory as "*some stakeholders can have interests that oppose each other*" [6] and change on a regularly basis, for example, as laws and standards are regularly updated, or due to knowledge gain of the customer and other stakeholders [12, 13]. Consequently, development artifacts have to be validated regularly against the current stakeholder intentions. In this context, validation means the "*confirmation by examination that requirements (individually and as a set) define the right system as intended by the stakeholders*" [14].

2.2 Behavioral requirements

In the embedded systems domain, the term "behavioral requirements" refers to an interaction-based specification of the necessary behavior that the system exhibits toward its environment (e.g., [4, 8, 15, 16]). The behavioral requirements specify a detailed and complete set of interaction sequences between the system and its context. The definition of all necessary interaction sequences in the behavioral requirements is important to ensure proper definition of the systems interfaces and ports (cf. [4, 15]). Interaction sequences define inputs to the system and outputs from the system as well as the timely order in which inputs and outputs are processed [16]. Message sequence charts (MSCs) and MSC-like languages are an adequate means to specify the behavioral requirements of embedded systems as they focus on the interaction-based behavior the system shall fulfill in the interplay with users and other systems. Consequently, this kind of languages is commonly used in industrial practice to specify interaction sequences in embedded systems' development [17]. In this paper, we use ITU MSCs as introduced by the recommendation Z.120 [18], which is an internationally binding standard issued by the International Telecommunication

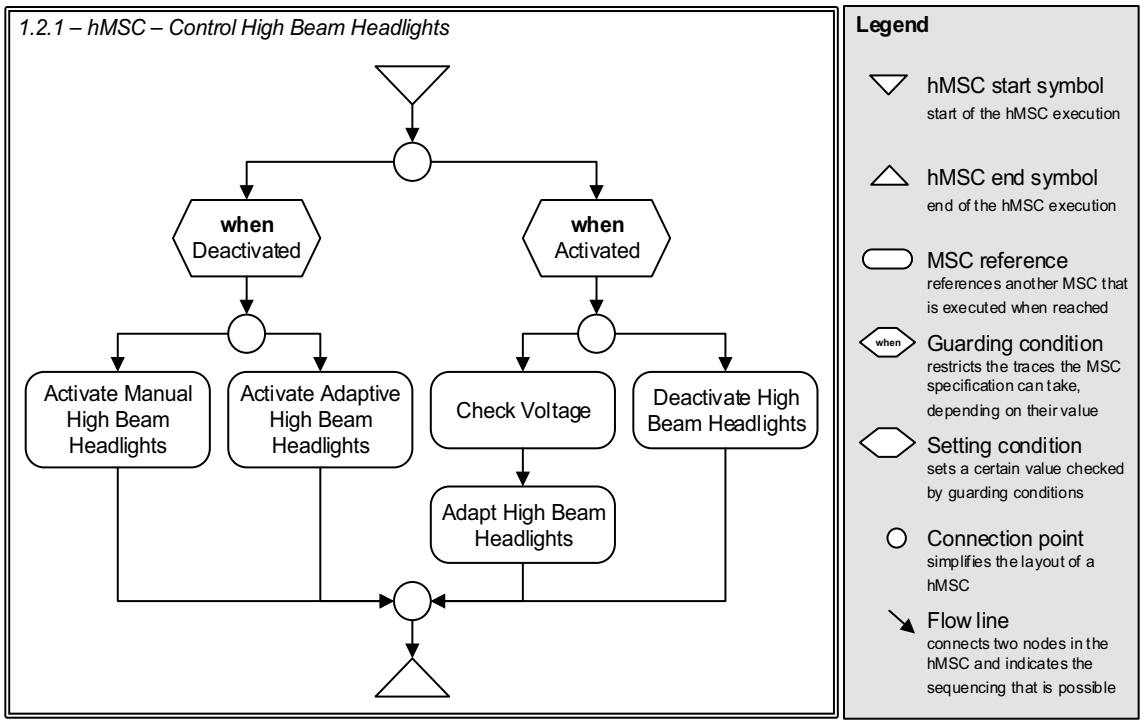


Fig. 1 Excerpt from the behavioral requirements of the ELS (hMSC—*Control High Beam Headlights*)

Union (ITU). ITU MSCs (in the following abbreviated as "MSCs") have been chosen for two major reasons:

a. Due to the safety–critical nature of embedded systems, the embedded industry is strongly concerned with adherence to regulations and international standards. Hence, an international standard issued and kept up-to-date by a regulatory body of the United Nations was seen by our industrial partners as more appropriate as other non-standardized languages.
b. MSCs are equipped with a diagrammatic means (i.e., high-level message sequence charts) for structuring different sequence diagrams. This is advantageous as the behavioral requirements shall specify a complete set of interaction sequences (although at a rather abstract level) the system will exchange with its context. Using MSCs the engineers are able to define a complete specification of the interaction-based behavior the system shall exhibit.

Throughout the remainder of the article, we use an automotive "adaptive exterior lighting system" (ELS) as running example. The specification of the ELS was developed in close collaboration with industrial partners to provide a model-based specification of a real-world embedded system for research purposes.

An MSC-based specification contains diagrams of two different types: "high-level message sequence charts" (hMSC) and "basic message sequence charts" (bMSC).

The hMSC diagrams structure the specification document. They define the order in which the bMSC diagrams are executed. Figure 1 shows an hMSC specifying the overall intended behavior of the high beam headlights.

The diagram shows several nodes (e.g., the start node, the end node, nodes referencing bMSC diagrams, connection points used to improve readability and conditions, which are used to restrict the control flow across all diagrams) and flow lines between these nodes. In the example, it is distinguished whether the high beam headlights have already been activated or not. If the high beam headlights have been activated, they can either be deactivated or automatically adapted to aid in the current environmental situation (e.g., if another car is approaching, the illumination area might be reduced). To conduct an adaption of the high beam headlights it is a prerequisite that the voltage is within predefined limits. This must be explicitly checked, since otherwise the light bulbs and other hardware parts might be damaged. If the high beam headlights are deactivated, either the manual high beam headlights or the high beam headlights providing the adaptive functionality can be activated.

The detailed behavior required to be performed by the system when a node is reached is defined by the complementary

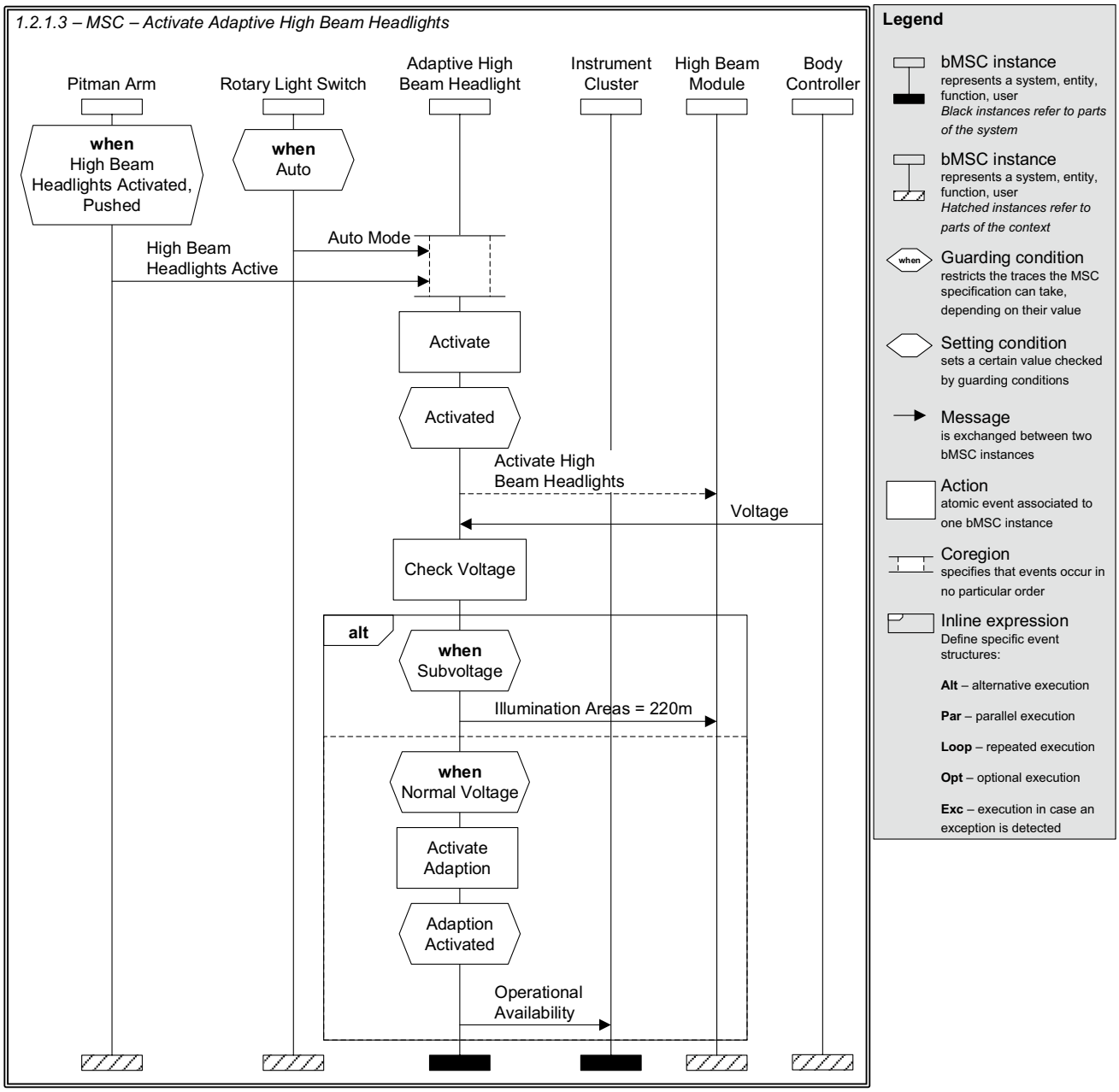


Fig. 2 Excerpt from the behavioral requirements of the ELS (bMSC—*Activate Adaptive High Beam Headlights*)

bMSC diagrams. Figure 2 presents the bMSC diagram for the node *Activate Adaptive High Beam Headlights* in the hMSC of Fig. 1.

As exemplarily shown above, a bMSC defines the interactions between several instances. In a bMSC, instances are either predefined parts of the system to be developed (depicted as solid black rectangles, i.e., *Adaptive High Beam Headlights* and *Instrument Cluster*) or users, systems, sensors and actuators in the context of the system to be developed (depicted as hatched rectangles, e.g., *Pitman Arm*, or *Rotary Light Switch*). Furthermore, conditions are annotated, and it is visualized when internal operations need to be conducted (e.g., to compare the actual voltage received from the body controller with the predefined voltage limits). The bMSC defines the desired interaction-based behavior to activate the adaptive high beam headlights: First, the pitman arm must indicate that the high beam headlights shall be active (this is most likely the case when the driver pushes the pitman arm), and the rotary light switch must indicate that it is in auto mode. Subsequently, the high beam headlights are activated, and the voltage is checked. If the voltage is lower than necessary, the illumination area of the high beam

module is set to a constant value. Otherwise, the instrument cluster shall indicate that the adaptive high beam headlights are activated.

2.3 Functional design

The *functional design* specifies system functions to be implemented, their hierarchical structure and each function's planned behavior (cf. [8, 19]). Thereby, it specifies interactions and dependencies between the functions in such a way that the interplay between different functions fulfills the behavioral properties documented in the behavioral requirements. The functional design is typically created based on the behavioral requirements and afterward further developed as other activities (e.g., the function partitioning) and artifacts (e.g., the electric and electronic architecture) provide further insights into the way the system under development must be shaped.

The functional design is typically documented using:

a. *static models* to specify the set of system functions as well as relevant functions provided by external systems together with the message exchange between functions;
b. *behavioral models* to specify each function's detailed behavior. The function's behavior model defines how outputs are generated depending on the respective inputs the function receive.

As a result, an interaction sequence specified by the behavioral requirements cannot be assessed in the functional design by investigating a single function, but must be evaluated by investigating the defined interplay between multiple functions cf. [7]. "Function network diagrams" define the system functions and their relations (i.e., the messages they exchange) on a structural level, and each function's detailed behavior is defined by a separate interface automaton. Like finite state machines interface automata [20] consist of states and transitions but additionally differentiate between inputs and outputs. Furthermore, interface automata are compositional (i.e., to combine the system behavior from the different function behaviors) and can be easily verified against a behavioral specification of the system's estimated runtime environment. Consequently, interface automata are a good means to be used for the specification of the functional design. This finding is substantiated by our industry partners from the automotive, avionics and automation industry, who preferred function network diagrams and interface automata as basic formalisms for a model-based specification approach for embedded systems' system and function networks (see [21]).

Figure 3 shows a function network diagram from the functional design of the ELS.

The function network diagram in Fig. 3 focuses on the sub-functions of the function *Control High Beam Headlights*. It shows the system's sub-functions (solid rounded rectangles; e.g., *Activate Manual High Beam Headlights*, or *Check Voltage*) and context functions (dashed rounded rectangles; e.g., *Move Pitman Arm*, or *Detect Vehicles*), which are provided by other systems and used by the system's sub-functions. In addition, the interactions between the functions are defined in terms of signal and message exchange (solid arrows). Dependencies between functions (e.g., enablement relations to define necessary partial execution orders of the functions), which are not reflected by direct interactions, are also defined (dashed arrows). As can be seen, the function to activate the adaptive high beam headlights has the same messages as input and output has previously been defined in the requirements (see Fig. 2).

Figure 4 shows two interface automata defining the behavior of the functions *Activate Adaptive High Beam Headlights* and *Check Voltage*.

As can be seen in the excerpt shown in Fig. 4, the function *Activate Adaptive High Beam Headlights* interacts with *Check Voltage* (concerning the message *Adoption Activated*). Therefore, these two system functions within the functional design must be evaluated together to decide whether the behavioral requirements from Fig. 2 are fulfilled by the functional design.

3 Problem statement

In this section we discuss the need for supporting the manual validation of behavioral requirements and functional design against stakeholder intentions (Sect. 3.1) and briefly outline potential solutions to discuss the applicability of currently existing approaches (Sect. 3.3).

3.1 Industrial need for manual validation

In function-centered engineering processes in the embedded systems domain, typically different engineers, organizational units or even suppliers are involved. Thus, artifacts are changed and updated many times during the development process as the stakeholder intentions about the system regularly change (e.g., [12, 13]).

Engineering processes in the embedded systems domain are typically scattered across a multitude of companies. Due to the very nature of engineering processes in the embedded systems industry, it cannot be ensured that stakeholder intentions are appropriately reflected within behavioral requirements and the functional design. In the automotive industry, typically, the original equipment manufacturer (OEM), its main suppliers (i.e., suppliers producing control units and their software or other major

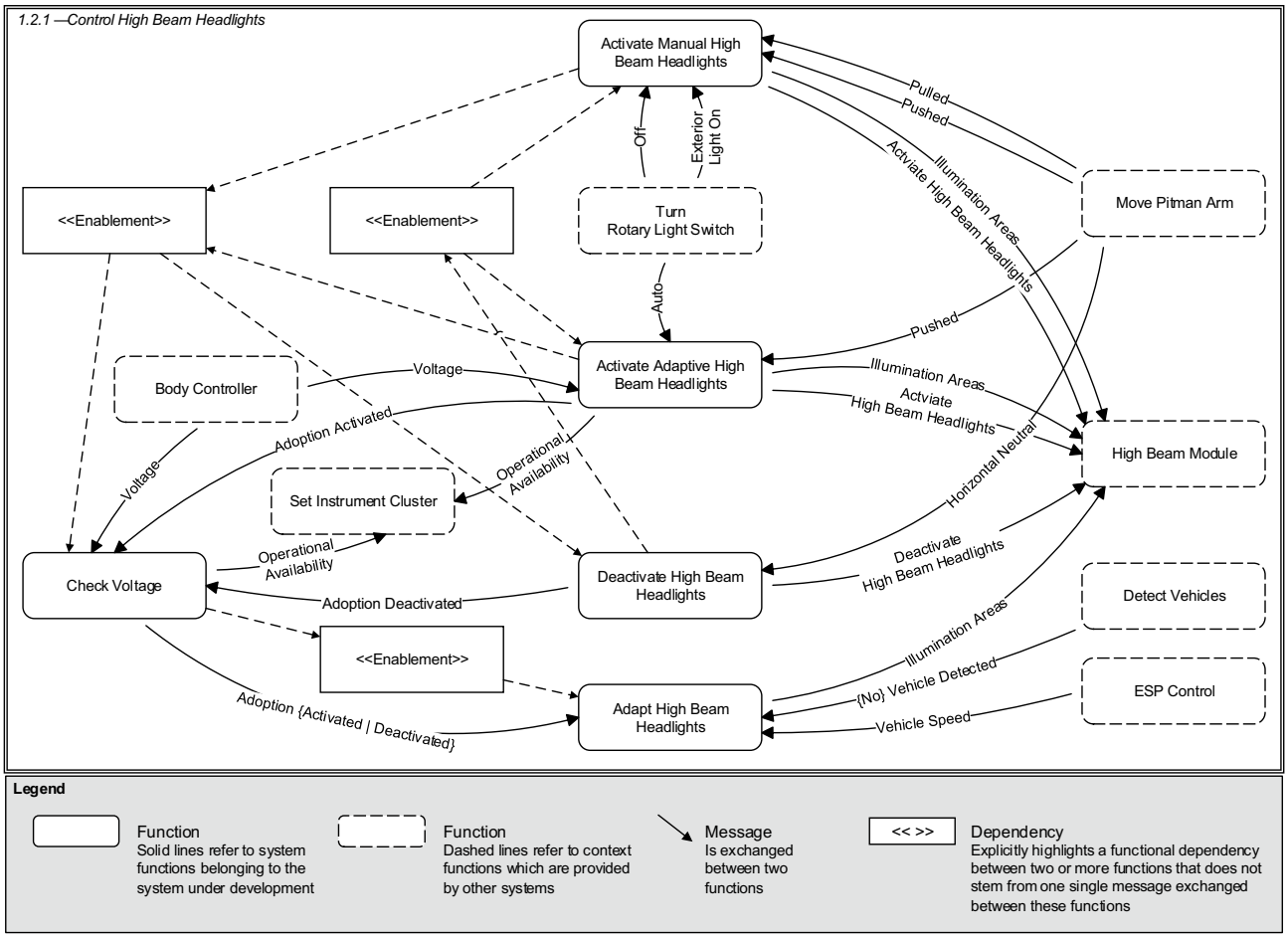


Fig. 3 Excerpt from the functional design of the ELS (function network diagram—*Control High Beam Headlights*)

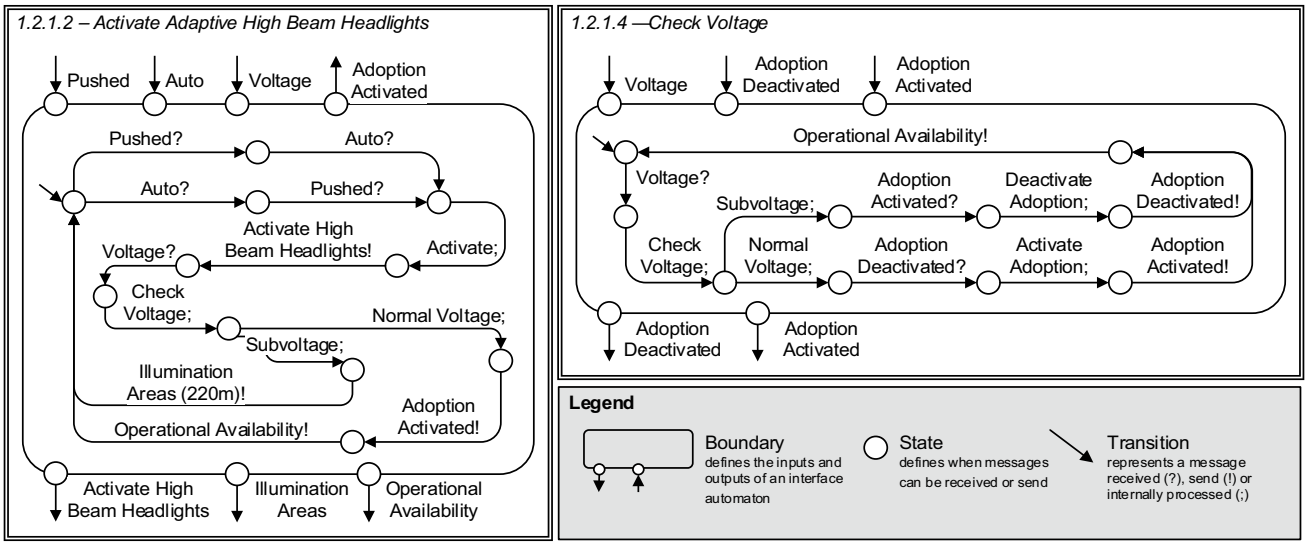


Fig. 4 Excerpt from the functional design of the ELS (interface automata—*Activate Adaptive High Beam Headlights* and *Check Voltage*)

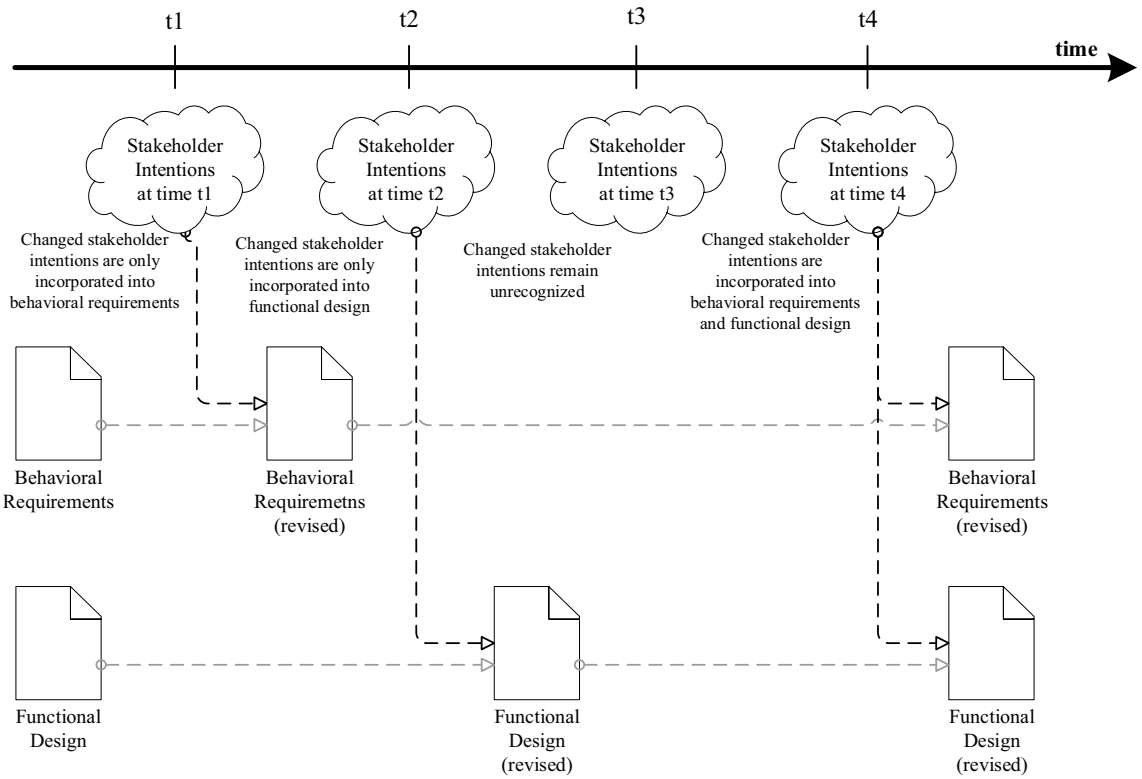


Fig. 5 Examples for revisions of behavioral requirements and functional design originated in changed stakeholder intentions

parts of a vehicle) as well as a multitude of low-tier suppliers (i.e., suppliers of the main suppliers and suppliers thereof) are involved in the development process for an individual system. The involvement of different companies in the development process also impacts the individual development processes within the companies. For instance, a major supplier needs to share and negotiate its requirements documents (i.e., the behavioral requirements) with the OEM. Hence, supplier and OEM negotiate some format and tool to be used. Depending on the market share of the supplier, the OEM might also simply advice the supplier to use a specific work environment. In contrast to the requirements, the functional design is driven from a reuse point of view. OEMs and major suppliers possess large function databases. For a development project, functions are taken from these databases and evaluated whether they fit the behavioral requirements or not. Commonly, functions do not fit perfectly, so either adapter functions are defined to integrate the original function within the new environment or the function is revised and later on stored back to the database as new function (cf. [9]). Hence, each supplier will use its own function database and therefore its own tooling environment for the functional design. As the behavioral requirements are defined in different tools, depending on the cooperation with the OEM, often no standard connectors allowing, e.g., end-to-end traceability exist.

Another issue impeding keeping behavioral requirements and functional design consistent emerges from the involvement of different teams within the individual companies. Depending on the organizational structure, teams may, for instance, be responsible for certain engineering disciplines (e.g., system requirements engineering, functional specification, development, test) or responsible for certain constituents of the overall system (e.g., software, mechanic, E/E architecture). In this context, it appears that, retrospectively, it is not traceable, which artifact has been changed first. Also, for instance, the requirements engineers might update the behavioral requirements but might incorporate necessary changes to the functional design only provisional. Function engineers will then rework the functional design to achieve a sound and well-formed functional design. However, it is not ensured that this revision correctly reflects the original intent of the behavioral requirements.

Therefore, there is a need to validate both specification artifacts against the stakeholder intentions. Figure 5 shows exemplarily how multiple changes of stakeholder intentions occurring over time lead to the situation that the behavioral requirements and the functional design do not appropriately reflect the current stakeholder intentions.

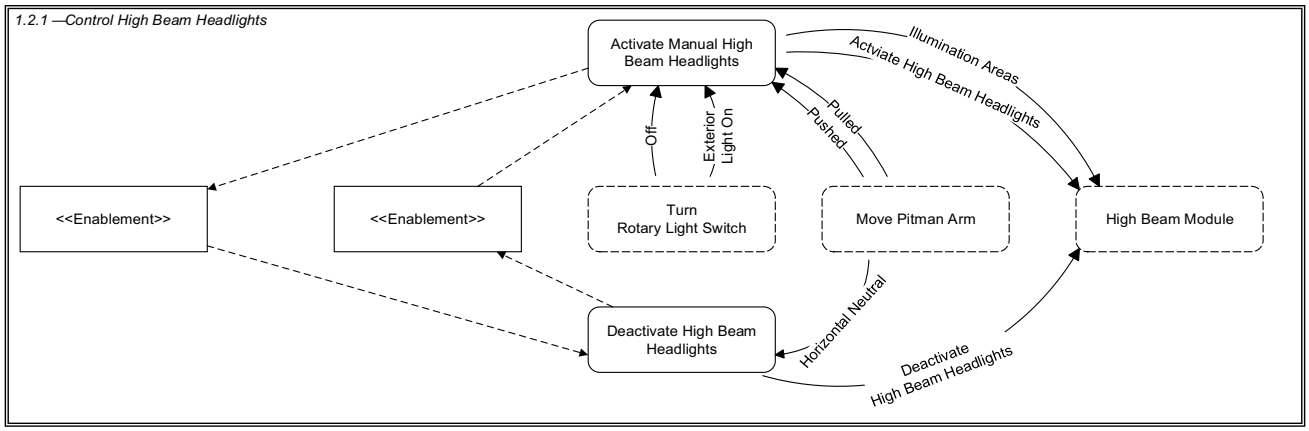


Fig. 6 Revised function network diagram within the functional design of the ELS

Some changes in stakeholder intentions are incorporated only into the behavioral requirements (for instance, in Fig. 5 at t1), while others are only reflected in the functional design (e.g., in Fig. 5 at t2). Furthermore, some intention changes are not recognized at all (at t3 in Fig. 5), while others are incorporated into both artifacts right away (for instance, at t4 in Fig. 5). Eventually, due to the different ways of handling changes in stakeholder intentions, the behavioral requirements and the functional design become inconsistent and invalid, i.e., both specification artifacts no longer reflect the current stakeholder intentions appropriately.

Furthermore, manual validation is commonly mandated by safety standards and quality assurance standards. For requirements [14] mandates manual inspections to validate *"that requirements (individually and as a set) define the right system as intended by the stakeholders"* and to verify *"that requirements (individually and as a set) are well formed."* Other development artifacts such as the functional design must also be subject to manual analyses, such as a failure mode and effects analysis (FMEA) or a functional hazard assessment (FHA) (cf. [22, 23]).

These are conducted as visual inspections of development artifacts trying to identify potential hazardous shortcomings resulting from the specification.

A further point for manual validation activities results from the fact that in today's engineering processes for embedded systems, stakeholder intentions are usually not sufficiently documented to allow for automated validation of engineering artifacts against the stakeholder intentions. The documented stakeholder intentions would need to be formalized to allow for automated checks, and it would have to be ensured that these documented stakeholder intentions are always up-to-date. While the former results in additional effort needed, the later only shifts the problem, as in this case, the documented stakeholder intentions would need to be manually validated regularly instead. Consequently, to be able to validate the functional design solely against the behavioral requirements, the validity of the behavioral requirements with respect to the stakeholder intentions must be ensured manually in the engineering process.

3.2 Motivating example

Let us assume that during development of the example ELS (see Sect. 2 for excerpts of the ELS' behavioral requirements and functional design) it is recognized that the specified adaption of the high beam headlights conflicts with certain governmental laws. These laws might, for example, have not been considered during the creation of the behavioral requirements. This can be due to management decisions of the original equipment manufacturer who originally did not intend the product to be sold in markets where these laws are in place. Hence, the functional design is updated in such a way that the functionality is no longer provided and the system complies with these laws.

In terms of stakeholder intentions this means: Originally it was intended that the high beam headlights automatically adapt their illumination range depending on the actual context situation, i.e., decrease illumination range when vehicle moving in counter direction is detected and increase illumination range when no vehicles are detected that could be blinded by an increased illumination range. This original stakeholder intention has been incorporated into the excerpts of behavioral requirements and the functional design shown in Sects. 2.2 and 2.3. Currently the stakeholder intentions have changed: Now, the stakeholders do explicitly not want the ELS to automatically increase and decrease the illumination range of the high beam headlights. Assume that due to this change of stakeholder intentions the functional design is updated. The corresponding revised function network diagram is presented in Fig. 6. As shown, the functional design of the system has been simplified since all functionality

solely needed for the adaptation has been removed from the diagram.[1]

At this point, the functional design has been updated, while the behavioral requirements have not. Such cases are common in industrial practice [9]. This can be, for example, due to spatial and organizational separations, where changes performed by one organizational unit on one artifact are not directly recognized by the organizational unit responsible for the other artifact. In the example of the ELS, this means that the behavioral requirements are no longer valid with respect to the stakeholder intentions and must be revised according to the current stakeholder intentions. Namely, the activation of adaptive high beam headlights and the adaption itself must be removed from Fig. 1. Additionally, the related bMSCs must also be removed from the behavioral requirements.

In many cases, such changes cannot be performed automatically. For instance, if another revision was made to the functional design, which does not correspond to a change in stakeholder intentions, this change must not be applied to the behavioral requirements automatically.

3.3 Potential solution approaches

Considering existing approaches for model verification and validation (see also Sect. 8), three general solution approaches for ensuring the validity of behavioral requirements and the functional design toward the stakeholder intentions are conceivable:

1. *One artifact is manually checked against the stakeholder intentions*, w*hile the other artifact is, subsequently, automatically verified (e.g., using model checking) against the former.* Considering current approaches and industrial practice, most likely the behavioral requirements would be validated against the stakeholder intentions, and the functional design would be verified against the behavioral requirements. However, in this case changed stakeholder intentions, which have yet only been incorporated into the functional design, are not considered. Hence, there exists the risk to overlook changed stakeholder intentions during the validation and verification process.
2. *Both artifacts are manually checked against the stakeholder intentions*. In this case all information documented in the artifacts under review is considered. However, it cannot be ensured that both reviews result in consistent behavioral requirements and functional design, as it cannot be assumed that two manual reviews result in the same defects found. Furthermore, the changed stakeholder intentions already documented in the functional design are also valuable for validating the behavioral requirements, and vice versa. However, in this scenario the latest versions of the behavioral requirements and the functional design, which have incorporated the latest known changes of stakeholder intentions, are not mutually used as reference during the review of the respective other artifact.
3. *Behavioral requirements and functional design are reviewed in a combined validation process*. In this case, the reviewers do not validate the behavioral requirements and the functional design separately, but both artifacts in a combined review. That way, the behavioral requirements can serve as reference in validating the functional design, and vice versa. This aids in the review of both artifacts, as, for instance, the reviewers can check whether a stakeholder intention correctly incorporated into the functional design has also been properly specified in the behavioral requirements.

While in case 3 both the behavioral requirements and the functional design can serve as reference for validation of the respective other artifact and thereby aid the review of both artifacts, some challenges still remain. Manual reviews, in particular perspective-based reading, have proved as an efficient and effective technique for validating single specifications (e.g., [24, 25]). However, the review of two specifications, which are closely related, increases the complexity of the review. This becomes challenging when considering error-prone aspects of manual validation. For example, changes of one artifact might imply changes to the other, which remain undiscovered, or defects might be detected in one artifact but are overlooked in the other.

4 Dedicated review model

To aid the manual review of embedded systems' behavioral requirements and functional design, we propose a dedicated review model that shall aid in this manual activity. The approach supports the manual validation from the perspective of requirements engineering by providing an automatically created review model, which integrates the information given in behavioral requirements and functional design. Figure 7 sketches the solution idea of the approach.

In the first automated step, the review model is created based on the behavioral requirements and the functional design. This review model enables the reviewers to conduct the review of behavioral requirements and functional design within a single, integrated review of the review

[1] To cope with the situation of different laws for different markets, commonly the use of variant management is suggested. Different variants of the product are created for each market. However, this is out of scope of this paper as the problem that behavioral requirements and functional design are inconsistent and do not correctly reflect the stakeholder intentions still persists.

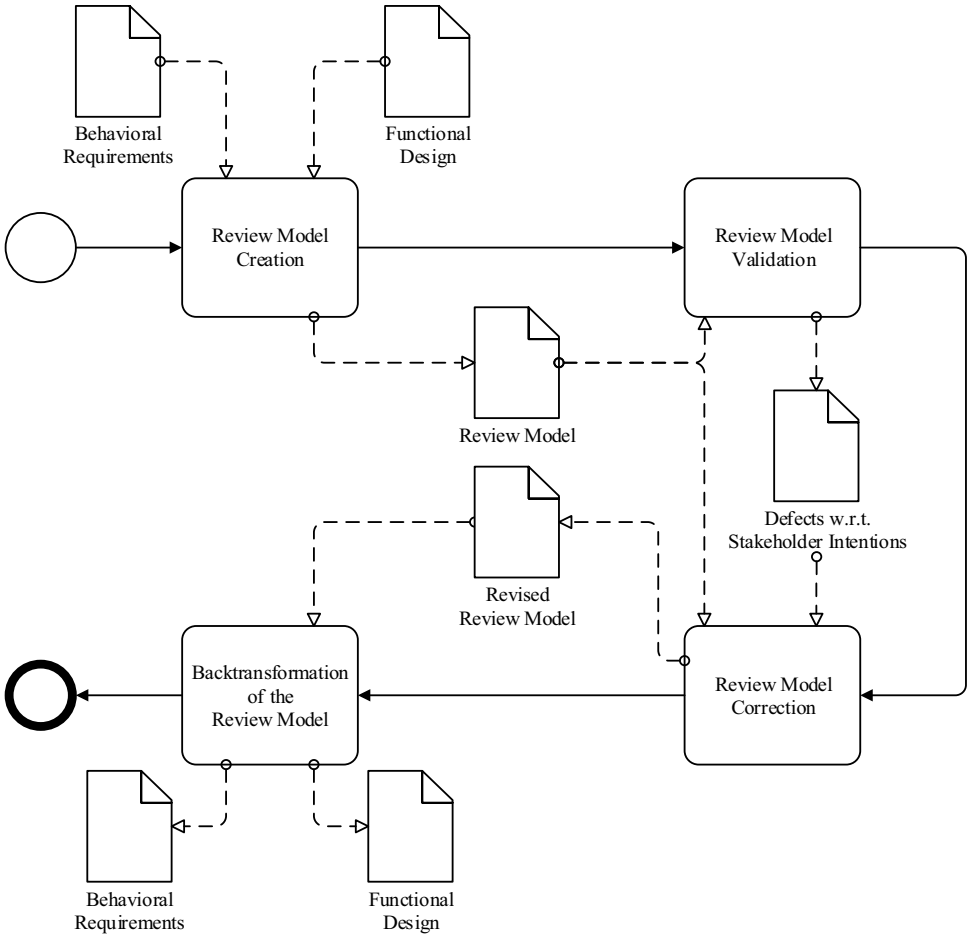


Fig. 7 Approach for the combined validation of behavioral requirements and functional design based on a dedicated review model

model. Different diagrams within this review model display behavioral properties that are: (1) consistent in both artifacts; (2) only specified in the behavioral requirement; or (3) only specified in the functional design.

Distinguishing these different kinds of diagrams structures the detection and correction of defects in the behavioral requirements and the functional design in a manual review. After the review, it is possible to either detect the defects in the original artifacts and correct them, or, as shown in Fig. 7, to correct the review model and automatically update and correct the original specifications based on the changes made in the review model.

4.1 Diagram types

The generated review model differentiates between interaction sequences consistently defined in both the behavioral requirements and the functional design, interaction sequences only defined in the behavioral requirements and interaction sequences only defined in the functional design. To aid the reviewer, we suggest the use of a review model in ITU MSC notation, consisting of an hMSC, which keeps the structure defined in the behavioral requirements, and bMSCs presenting the interaction sequences for manual review. The bMSC diagrams within the review model can be differentiated into three categories:

- *Refinement diagrams* These diagrams depict specified behavioral properties, which are consistently defined within the behavioral requirements and functional design. Typically, these properties are correct, but still a reviewer will have to check the diagrams to eliminate the possibility that a behavioral property not desired by the stakeholders is consistently documented in both artifacts.
- *Unrefinable diagrams* These diagrams contain behavioral properties, which are specified within the behavioral requirements, but are not specified in the functional design. Therefore, the reviewers analyzing the review model will have to decide whether the behavioral properties must be removed from the behavioral requirements, or whether the functional design must be updated accordingly.
- *Unspecified diagrams* These diagrams depict behavioral properties, which are only documented within the functional design, but not specified within the behav-

ioral requirements. In this case, the reviewers analyzing the review model will have to decide whether the behavioral properties must be removed from the functional design, or whether the behavioral requirements must be updated accordingly.

The behavioral requirements and functional design are not intended to specify the exact identical concerns just using different modeling languages. In fact, the behavioral requirements place emphasis on the intended interaction-based system behavior, which is recognizable at the system boundaries. The functional design places emphasis on the definition of technically related functions, their behavior and the interplay of functions to achieve desired system functionalities and behaviors. Although the functional design is still an implementation neutral abstraction, which is not to be confused with logical architectures (e.g., the functional design typically does not define which functions are to be implemented in software and which functions shall be realized by hardware), it provides more details about the internal behavior of the system. For the definition of unspecified diagrams this means it must be taken into account that the functional design provides more behavioral traces, which in principle could lead to a huge number of unspecified diagrams. However, we do not compare the internal behavior of the system but the behavior that is recognizable at the systems boundaries. Regarding this externally recognizable behavior in the functional design it is explicitly mandated that it adheres to the behavior specified in the requirements. Hence, no additional external behavior may arise from the functional design that has not been already considered in the behavioral requirements—although with a different purpose-specific abstraction.

This limits the number of unspecified diagrams to be derived. Nevertheless, comparing different levels of granularity between behavioral requirements and functional design also means that refinement relations must be taken into account. For instance, the behavioral requirements might specify a certain message to be exchanged, which is refined in the functional design in more fine-grained messages as certain parts of the original message are processed by different functions. To cope with this issue, it must be assumed that some kind of refinement matrix, e.g., a data dictionary, exists that defines which behavioral requirements message has been refined into which functional design messages. However, it must be kept in mind that the erroneous or missing incorporation of changes of stakeholder intentions will most likely also have negative effects for such a refinement matrix.

In contrast to the bMSCs of the original behavioral requirements, the bMSCs of the review model also display the interaction sequences as specified by the functional design. Hence, the bMSCs are typically more detailed due to the more fine-grained nature of the functional design. Additionally, the bMSCs do not display the original instances as defined in the behavioral requirements. In the review model, the instances represent the functions, which have been defined in the functional design. This allows the reviewer not only to investigate whether all required messages are exchanged and whether the order of messages exchanged is correct, but also whether the messages are exchanged between the correct functions.

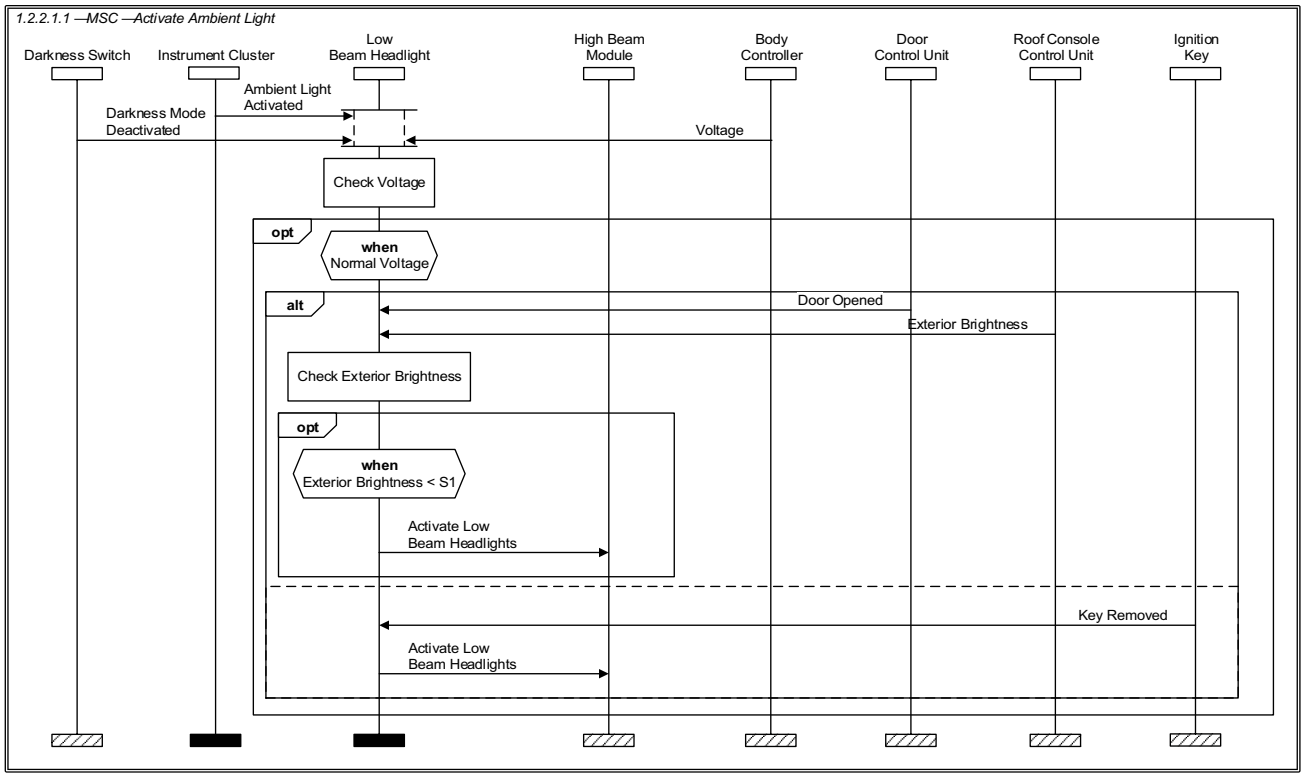
Figure 8 visualizes these differences between a bMSC of the behavioral requirements and a bMSC of the review model, exemplified by a refinement diagram.

Figure 8a shows a bMSC of the behavioral requirements, which defines the intended behavior to activate the low beam headlights as part of the ambient light to aid in entering and leaving the car in darkness. Figure 8b shows a function network diagram of the functional design depicting the specified functions and messages exchanged in between to activate the ambient light (the messages and functions that are involved to display the functionality intended by the behavioral requirements are shaded gray). Figure 8c shows the refinement diagram of the review model, which displays the same behavior as defined in Fig. 8a under consideration of the functional design shown in Fig. 8b. Figure 8c is a refinement of Fig. 8a. In particular, the messages exchanged with external instances are contained in both diagrams. The refinement bMSC in addition displays internal messages between system functions defined in the functional design. Furthermore, the refinement diagram of the review model does not display the original instance names; this information has been replaced with the function names as specified in Fig. 8b.

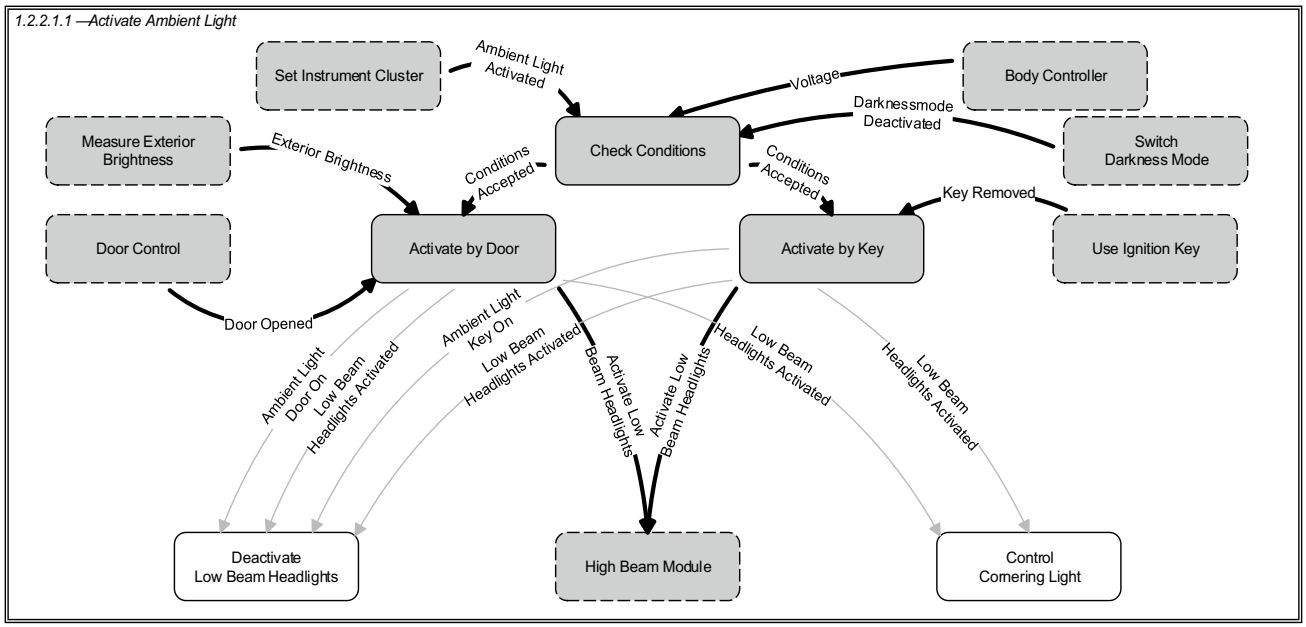
4.2 Automated generation

The review model can be generated by combining and harmonizing existing model transformation approaches and automata theoretic comparison operators. To generate the review model, behavioral requirements and functional design must first be compared and from the result the review model can be generated. Therefore, both artifacts must be first transferred into a comparable notation. We decided to use interface automata as the functional design already uses interface automata and MSCs are not only easily transferrable into automata notation but can also be generated from automata notation. Hence, a three-step approach leads to the generation of the review model:

- *Step 1* Behavioral requirements and functional design are transferred into interface automata to allow for comparison of both artifacts.

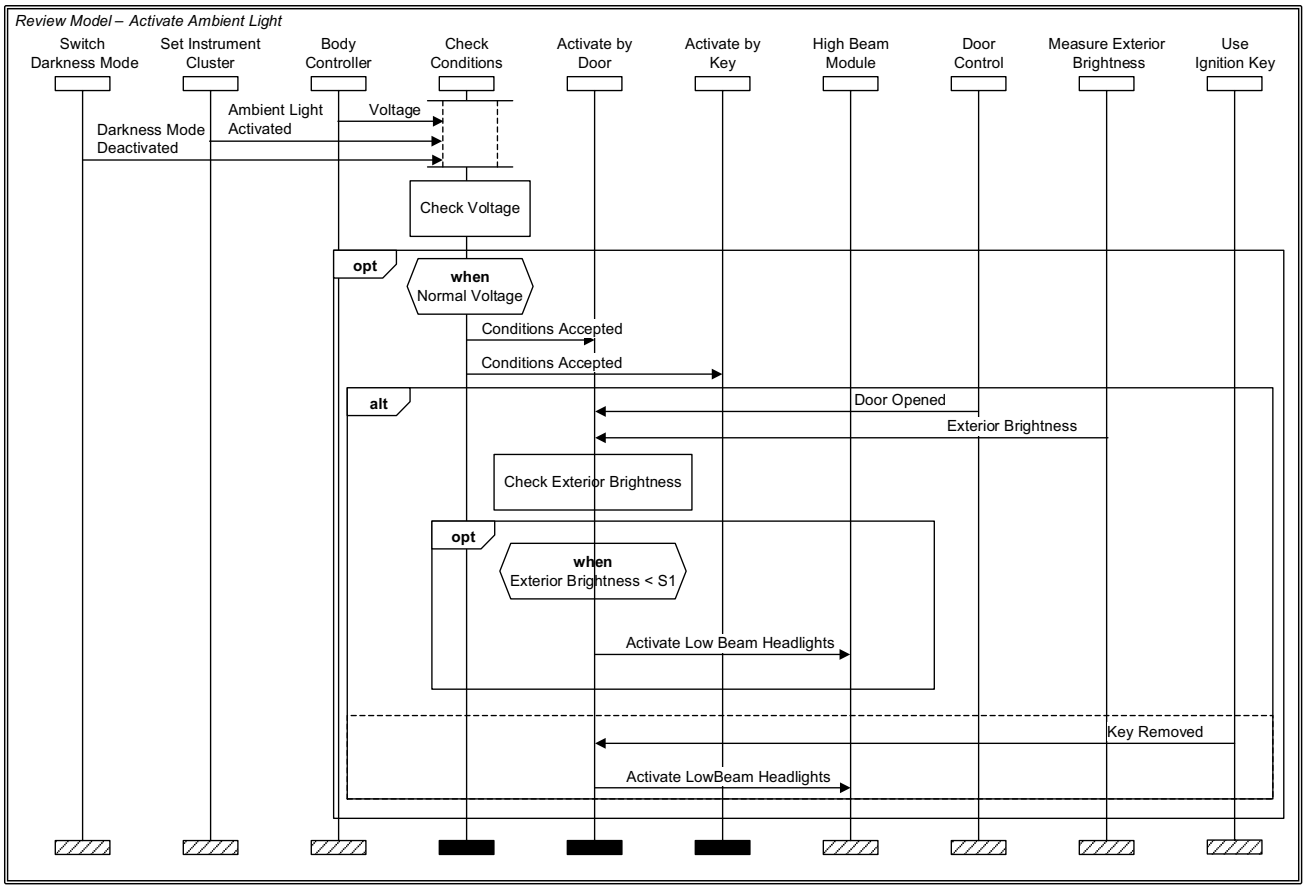


(a) bMSC of the Behavioral Requirements



(b) Function Network Diagram of the Functional Design

Fig. 8 Exemplary refinement diagram of the review model (**c**) and the corresponding diagrams of behavioral requirements (**a**) and the functional design (**b**)



(c) Resulting Refinement Diagram of the Review Model

Fig. 8 (continued)

- *Step 2* The generated interface automata are used as intermediate model to compare behavioral requirements and functional design.
- *Step 3* The review model is generated from the intermediate interface automata.

As our approach relies on existing approaches from the related work, which are combined, adapted to fit for MSCs and interface automata, and if necessary enhanced, we will not go into detail about the technical approach within this article. However, we provide the technical implementation of the used model transformations in combination with further experimental materials as online supplement: https://doi.org/10.5281/zenodo.1744665. In this article, we will refer to the original approaches we used and adapted and give some insight into the review model creation to allow proper understanding of the experimental setup reported in Sect. 5. Section 4.2.1 gives a more detailed overview of the approach and outlines, which existing approaches have been combined. Subsequently, Sects. 4.2.2, 4.2.3 and 4.2.4 will give some insight into the single steps of the approach.

4.2.1 Overview

The three steps for review model generation can be refined into more fine-grained steps. Figure 9 illustrates the combination of these fine-grained transformation and comparison steps needed to create the review model based on the behavioral requirements and the functional design.

First, behavioral requirements and functional design are prepared for comparison in Steps 1.1–1.4. In this phase intermediate models are created, which are used later on for comparing the behavioral requirements and the functional design. As intermediate models, interface automata were chosen, since these are already part of the functional design and provide the needed comparison operators:

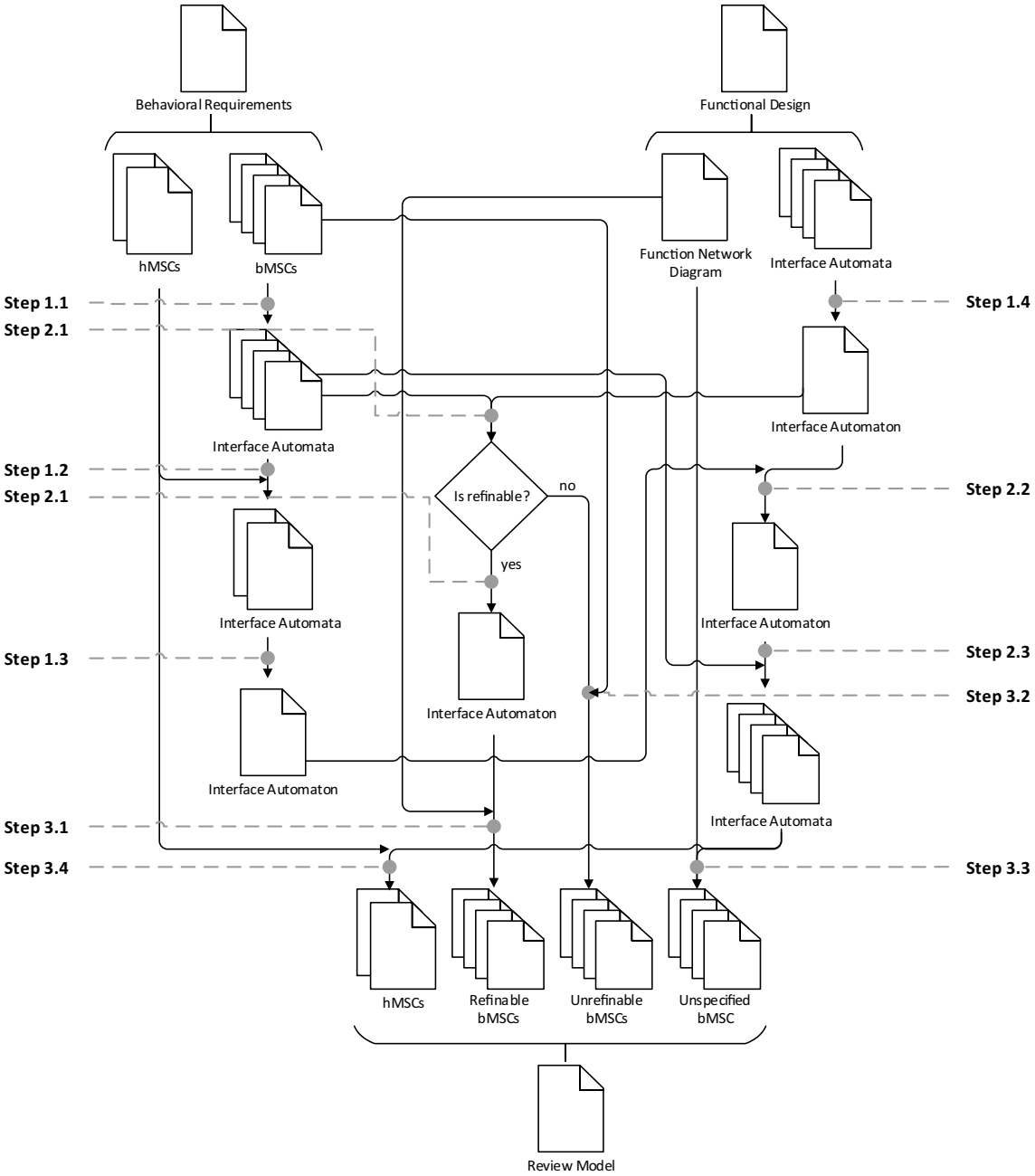


Fig. 9 Systematic generation of the review model by combining different model transformation and comparison steps

- *Step 1.1* For intermediate model generation, the bMSCs of the behavioral requirements are transferred into interface automata. For the transformation of bMSCs into automata notation a variety of approaches exist. We build our approach on the approaches by Alur et al. [26], Uchitel et al. [27–29] and Whittle and Jayaraman [19].
- *Step 1.2* To enable a comparison of all specified interaction sequences, the created interface automata from Step 1.1 are concatenated according to the hMSC using common concatenation operators for automata, i.e., [30]. Note that this concatenation is in accordance with the approaches used in Step 1.1 that consider hMSCs. However, approaches for synthesis of state-based behavioral models (e.g., modal transition systems, finite state machines) from interaction-based behavioral models (e.g., MSCs, life sequence charts, UML sequence diagrams) often do explicitly assume that there is no overall structure giving element such as the hMSC. These approaches synthesize a state-based behavioral model from unrelated and redundant models

as is often assumed in the context of scenario specifications (see, e.g., [31]). However, this is not necessary for the transformation of the behavioral requirements, as the hMSC defines the ordering of the bMSCs and thereby a global order is given and must not be retrieved.

- *Step 1.3* If the behavioral requirements define more than one system instance, which can happen if already in the requirements engineering phase major subcomponents to be implemented have been identified, in Steps 1.1 and 1.2 interface automata for all system instances have been created. These instances must now be composed, which is achieved using the original composition operator of interface automata [20].
- *Step 1.4* On the side of the functional design, the interface automata defining each function's behavior are composed to generate one overall automaton containing all specified interaction sequences using the composition operator of interface automata [20].

Second, the comparison of the behavioral requirements and the functional design is conducted using the created interface automata as input for Steps 2.1–2.3:

- *Step 2.1* The interface automata created from each bMSC are compared with the overall interface automaton of the functional design to investigate whether the defined interaction sequences are also specified (most likely in a refined manner) in the functional design. If this is the case, an automaton is created to define the refined interaction sequences. To do so, two comparison operators are needed. On the one hand, an operator is needed to determine whether two interface automata are equivalent in the sense of [32, 33]. On the other hand, an operator is needed to identify

  a. if an interface automaton is part of another interface automaton in the sense of [34] (i.e., whether the set of transitions of one automaton are a subset of the transitions of the other interface automaton and the orders in which those transitions can be executed are identical) and
  b. if an interface automaton describes a subset of the language of another interface automaton in the sense of [32] (i.e., whether the end-to-end paths defined by one interface automaton can be found in the other one).

- *Step 2.2* The interaction sequences only defined in the functional design are determined by calculating the difference between the automaton of the functional design and the corresponding automata of the behavioral requirements.

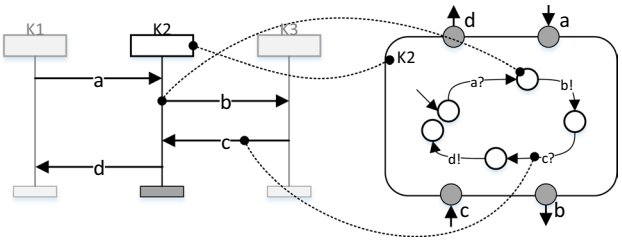


Fig. 10 Basic transformation relations between bMSC and interface automata

- *Step 2.3* The resulting automaton from Step 2.2 is analyzed for interaction sequences already defined by an existing bMSC using the operators from Step 2.1. These sequences are removed from the automaton, resulting in several interface automata defining interaction sequences, which are not defined by the behavioral requirements.

Third, after the comparison and definition of interface automata to be displayed as bMSC in the review model, the review model is generated from these intermediate models in Steps 3.1–3.4:

- *Step 3.1* The interface automata describing refinable bMSCs are transferred into bMSCs. As [18] defines automata semantics for MSCs, this can be achieved by applying the approaches from Step 1.1 in a reverse manner.
- *Step 3.2* The unrefinable bMSCs of the behavioral requirements are copied into the review model from the behavioral requirements specification.
- *Step 3.3* The interface automata defining interaction sequences only specified in the functional design are also transferred into bMSCs.
- *Step 3.4* An hMSC is created based on the behavioral requirements' hMSC, which is enhanced with references to the unspecified bMSCs specifying interaction sequences only defined in the functional design.

In the following, we will outline the steps from Fig. 9, focusing on how approaches taken from the state of the art are used, and how they are combined and aligned to fit in our approach for review model generation.

4.2.2 Creation of intermediate interface automata

Model transformation techniques are applied to transfer MSCs into interface automata (Steps 1.1–1.4 in Fig. 9). Model transformations and syntheses have been defined for the transfer of message sequence charts into finite state machines (e.g., [19, 29]), which are in conformance with the automata semantics defined by [18] for bMSCs. Based on

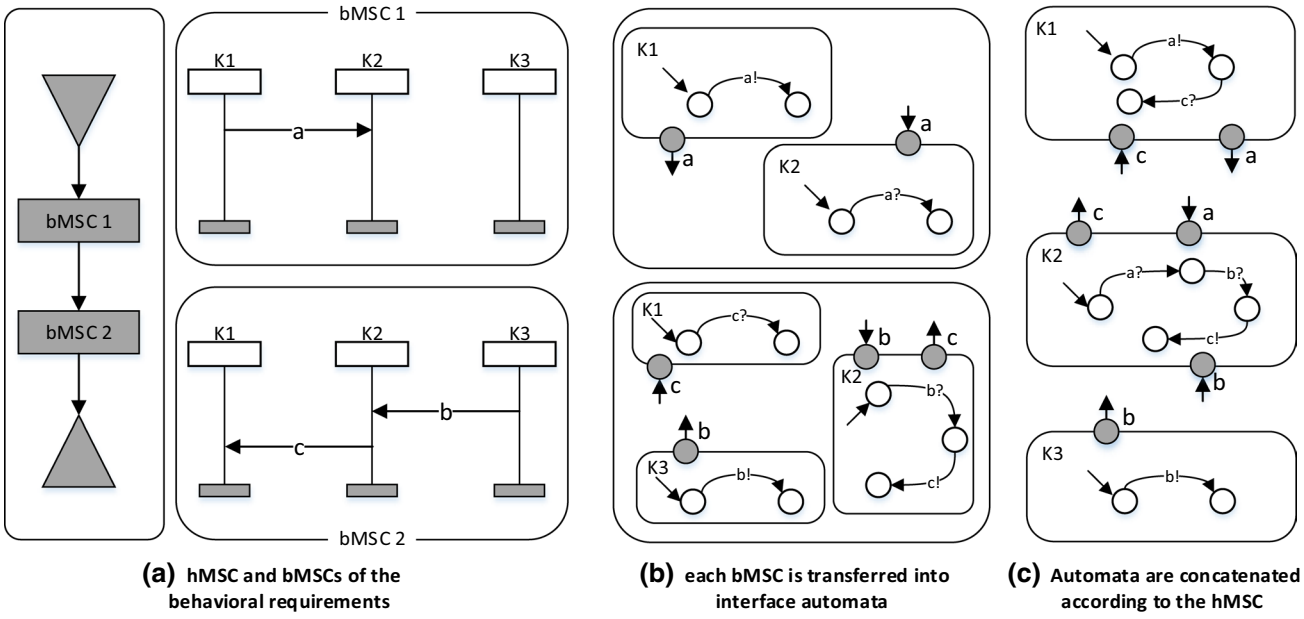


Fig. 11 Exemplary transformation of the behavioral requirements into interface automata

these approaches, in our previous work we have defined a model transformation into interface automata, which allows for automated comparison of different MSC specifications regarding their specified input/output behavior [35].

Basically, for each system instance defined by a bMSC the transformation derives one interface automaton. In this interface automaton, the behavior defined for the system instance transferred is described as originally in the bMSC. Figure 10 illustrates the basic transformations.

The instance K2 is transferred into the interface automaton. In this interface automaton, the messages sent and received by K2 are related to the transitions of the automaton. In addition, the states of the automaton are defined such a way that the same order of interaction sequences (w.r.t. K2) is described as in the bMSC.

Figure 11 gives an abstract example of the overall transformation of behavioral requirements into intermediate interface automata.

First, as outlined above, for each instance belonging to the system an interface automaton is created. This is shown in Fig. 11b: As can be seen, for bMSC1 two automata describing the behavior of K1 and K2 are derived; for bMSC2 three automata describing the behavior of K1, K2 and K3 are derived. Note that these automata, of course, only describe the behavior of the instances as defined by the original bMSC in Fig. 11a. Hence, in the next step the individual interface automata describing behavior of the same instances are concatenated as shown in Fig. 11c to describe the complete behavior of an instance, which has been defined in the behavioral requirements.

Finally, in case the behavioral requirements do not specify a single monolithic system instance but more instances belonging to the system, the automata describing the single system instances can be composed using the composition operator from [20] to represent the entire system behavior defined by the behavioral requirements.

Composition must also be used to prepare an interface automaton describing the entire system behavior specified by the functional design. To this end, the single automata describing each function's behavior are composed to a single automaton.

4.2.3 Comparison of intermediate interface automata

The comparison of the intermediate interface automata obtained from behavioral requirements (Steps 2.1–2.3 in Fig. 9) and functional design mainly relies on standard automata operations such as the difference operator from [30], which have been transferred and adapted to fit for the subtraction of interface automata.

Additional comparisons can be implemented by combining already existing operators of [20]. For example, an entire interaction sequence from the functional design, beginning from system start to system termination, might not be entirely described in the behavioral requirements. However, parts of this overall sequence are likely to be defined in bMSC diagrams. Hence, not only the overall behavior defined by behavioral requirements and functional design must be compared, but also the behavior defined by single bMSC diagrams must be identified within the overall

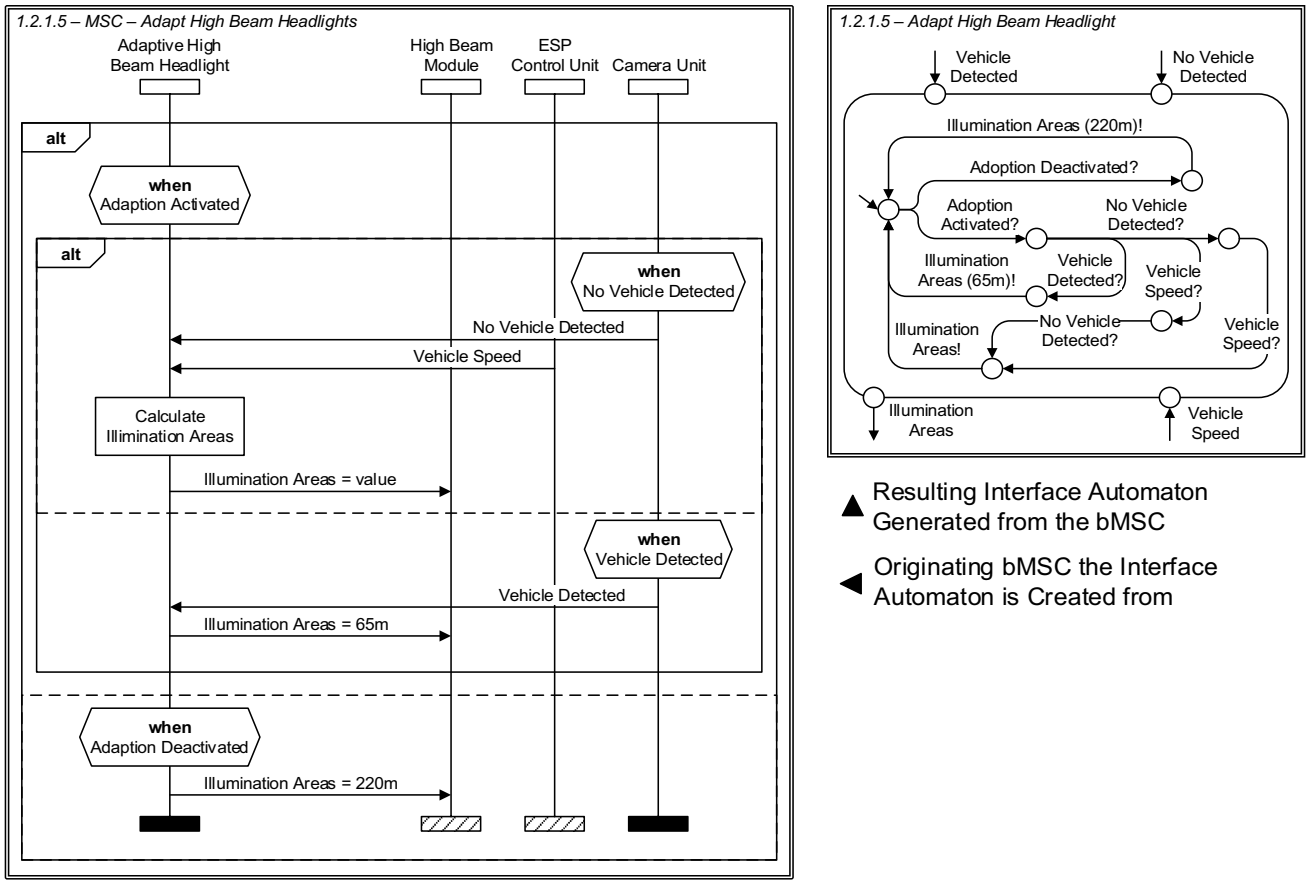


Fig. 12 Creation of an interface automaton for the instance *Adaptive High Beam Headlight*

behavior of the functional design. Therefore, it is of interest whether an interface automaton is part of another. In particular, two comparisons must be considered: It needs to be determined whether an interface automaton (a) describes traces that are also defined by another interface automaton (cf. [32]) and (b) describes an adhesive collection of states and transitions that are also defined in another interface automaton (cf. [34]).

4.2.4 Creation of the review model

As ITU recommendation Z.120 [18] defines the semantics of bMSCs using automata semantics, automata such as the interface automata can be easily transferred into bMSCs (Steps 3.1–3.4 in Fig. 9). The transformation is similar to the generation of the intermediate interface automata from bMSCs, but is performed in the opposite direction. In addition, as outlined in Sect. 4.1, the instances displayed in a bMSC are derived from the function network diagram. This is already shown in Sect. 4.1 in Fig. 8. Hence, the original bMSCs of the behavioral requirements and the refinement diagram of the review model differ in granularity as well as the instances contained. As discussed above the comparison between behavioral requirements and functional design is based on the externally recognizable behavior the system possesses at its interfaces. Hence, the more fine-grained level of detail of the functional design does not lead to the detection of deviations from the interaction sequences defined in the behavioral requirements, unless, of course, the internal behavior (i.e., the behavior of each function and the interplay between system functions) leads to an externally recognizable behavior at the system boundaries that is not in accordance with the behavioral requirements.

We will first exemplarily illustrate the transformation of a bMSC of the behavioral requirements into an interface automaton and subsequently the back transformation of the same interface automaton into a corresponding bMSC of the review model. Figure 12 illustrates a bMSC of the ELS and the corresponding, derived interface automaton of the system instance *Adaptive High Beam Headlights*.

The interface automaton defines all messages received and sent, as well as their possible execution orders, which result from the causal ordering in the bMSC.

Conducting the transformation as outlined in Fig. 12 backward will not result in the original bMSC used. This

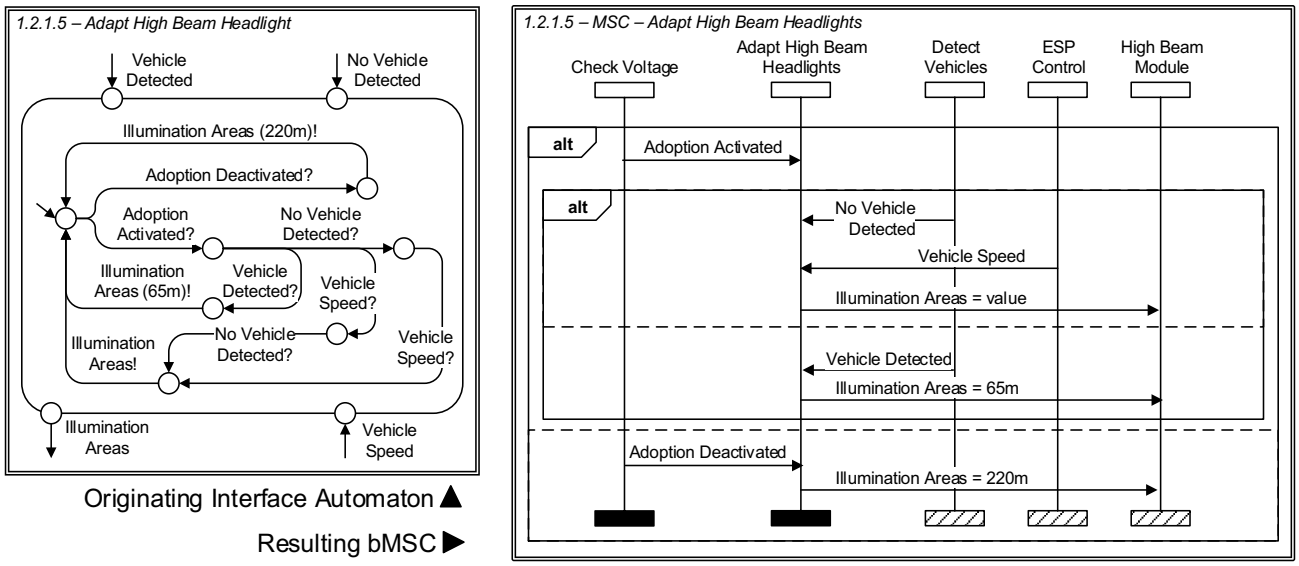


Fig. 13 Creation of a message sequence chart from an interface automaton

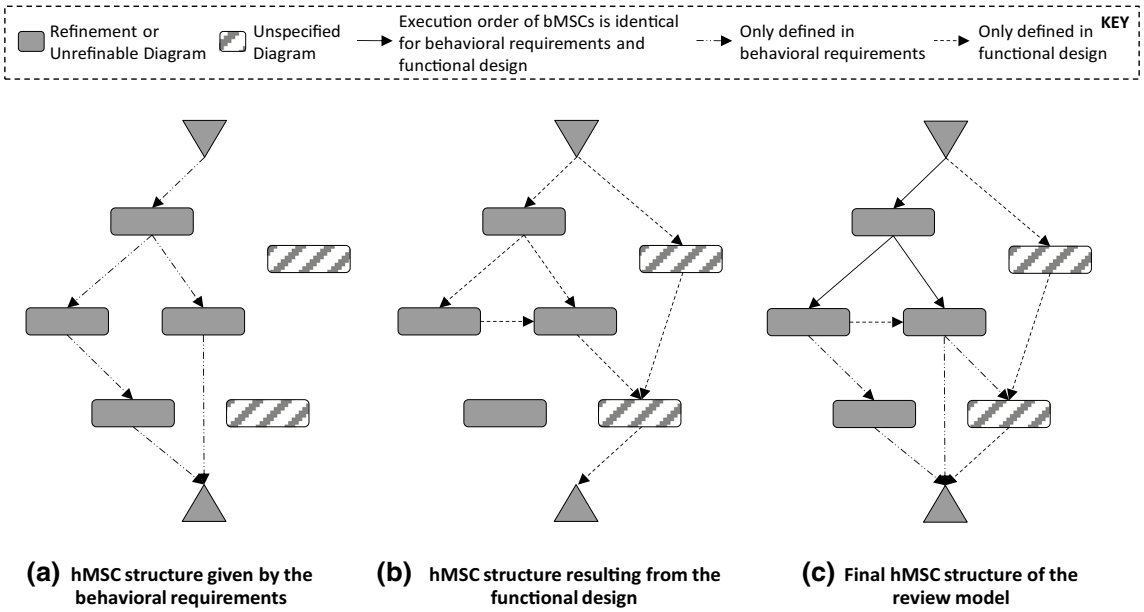


Fig. 14 Exemplary definition of the review model's hMSC structure

is due to information represented in the bMSC but not contained in the automaton. In addition, the resulting bMSC is more fine-grained in nature, as it details the single functions defined in the functional design. The result of the transformation of the automaton from Fig. 12 into a new bMSC is shown in Fig. 13.

As can be seen, the instances of the bMSC from Fig. 13 do not represent other systems that have been identified during requirements engineering, but do now represent concrete functions, which have been defined in the functional design. For instance, in contrast to Fig. 12 the bMSC from Fig. 13 uses a context system function to detect vehicles, which informs the ELS whether a slow driving vehicle has been detected or not. In the bMSC from Fig. 12 it was only noted that camera unit will somehow have to provide the relevant information.

Once the bMSC diagrams of the review model have been created, these diagrams must be referenced in an hMSC in such a way that the reviewers are able to investigate whether the specified interaction sequences of the bMSC are defined

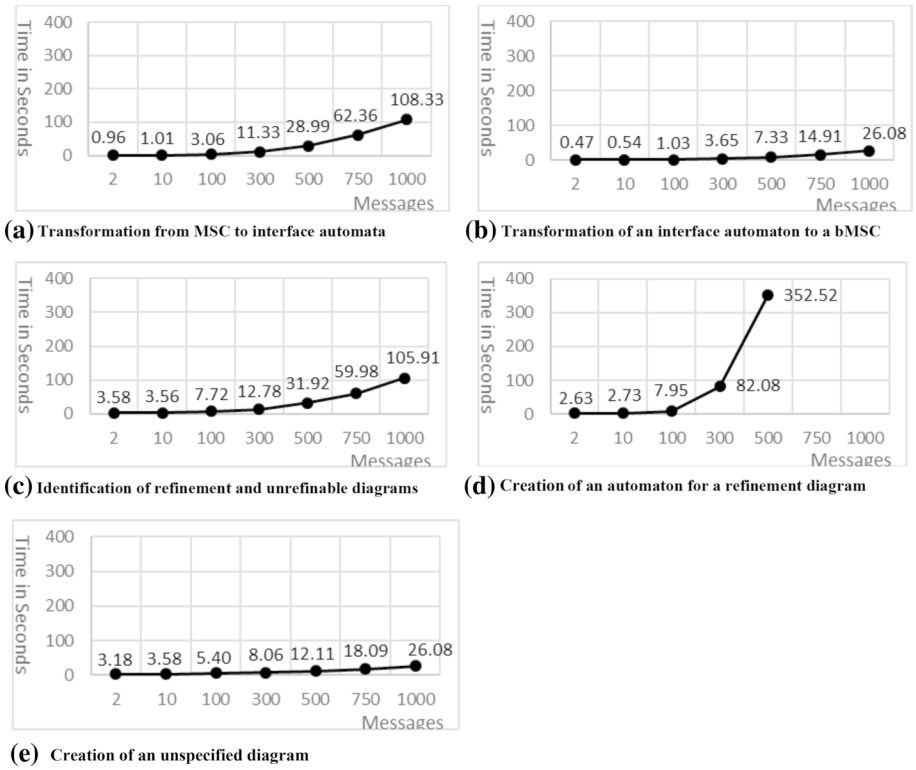


Fig. 15 Runtime measuring results of prototypical implementation

in the right chronological order. For instance, the initializing sequence of the ELS as defined in the functional design might be correct in terms of stakeholder intentions, but if initializing requires the ELS to be already initialized, obviously the initializing sequence is not ordered according to the stakeholder intentions. hMSC integration allows the reviewer to detect such cases. Figure 14 sketches the basic approach for hMSC integration.

Figure 14a shows the hMSC structure, which is given by the behavioral requirements. As can be seen, only refinement and unrefinable diagrams can be ordered, since the unspecified diagrams only originate from the functional design. Figure 14b shows the orderings of the bMSCs, which can be resolved from the functional design. In summary, Fig. 14c integrates both orders and highlights the differences in the review model depicted as defined in the key of Fig. 14.

Note that the functional design does neither provide an ordering relation like the hMSC of the behavioral requirements nor does the structure of the bMSCs come derived from the functional design naturally. The behavior defined by the functional design could in principle be divided into a variety of different possible bMSCs. For instance, a very simplistic functional design consisting of one function, which receives one message and sends another, can be either transferred into one bMSC showing the receiving and sending or into two bMSCs, one bMSC for each message. To cope with this complexity of combinations and to allow for a readable hMSC structure, the hMSC of the review model is mostly defined by the hMSC of the behavioral requirements. bMSCs resulting from the functional design are then integrated into this hMSC. These bMSCs are defined by comparison with the behavioral requirements. First the traces (i.e., end-to-end traces of the behavior) that are only defined in the functional design are identified. Then, using the comparison operators from Sect. 4.2.3, parts of these traces that are already defined by existing bMSC are identified and removed. The remaining adhesive parts of a trace are then transferred into a bMSC each, and these are ordered into the hMSC structure as sketched in Fig. 14.

4.2.5 Implementation

The proposed approach for review model generation was prototypically implemented and analyzed. The necessary transformations were implemented using QVTo [36], a commonly used model transformation language, which is part of the Query/View/Transformation Language (QVT [37]) defined by the OMG. The implementation of the QVTo model transformations can be accessed at: https://doi.org/10.5281/zenodo.1744665. Additionally, a Java program was developed to execute the different transformations in the correct order and handle the inputs and outputs. For the graphical representation of the diagrams we used Papyrus, as the examples were provided in the respective files by our industry partners. Runtime analyses show that review models can be automatically created in finite and acceptable

Table 1 Overview of the controlled experiments

Experiment	Description
Experiment 1	The experiment compares *effectiveness*, *efficiency*, *user confidence* and *subjective supportiveness* of *graduate students* reviewing the review model with the review of behavioral requirements and functional design
Experiment 2	The experiment compares *effectiveness*, *efficiency*, *user confidence*, and *subjective supportiveness* of *undergraduate students* reviewing the review model with the review of behavioral requirements and functional design. The experimental material as well as the entire experimental setup is identical to experiment 1, apart from the experiment participants. The use of participants with different experience and skill levels in both experiments allows investigating whether the experiences and skill level impacts the review
Experiment 3	The experiment compares *effectiveness*, *efficiency*, *user confidence* and *subjective supportiveness* of *graduate students* reviewing the review model with the review of the functional design. In contrast to experiments 1 and 2, in this experiment the user does not review both the behavioral requirements and the functional design for comparison with the review of the review model. This allows investigating the influence of the chosen notation format on effectiveness, efficiency, user confidence and subjective supportiveness of the review

time. Measures were taken on a conventional laptop with a 2.5-GHz dual-core processor and 4-GB RAM using Eclipse Luna on a Windows 10 Professional (64 bit) operating system and JRE8u60. As input we used randomly generated message sequence charts and interface automata with different levels of complexity. These diagrams were of larger size and higher complexity as real diagrams in practice. For each size, we tested ten different diagrams and ran the transformation 100 times to reduce the influence of possible irregularities. The results represent the average time one transformation took. Figure 15 shows runtime measuring results for the different steps in model generation.

Except for Fig. 15d, all parts terminate in acceptable time even for large diagrams. As can be seen the calculation of a refinement diagram takes exponential time. Nevertheless, it takes on average 352 s (less than 6 min) for a diagram with 500 messages, which can be considered acceptable. As the ITU standard explicitly allows for the separation of bMSCs, this size is unlikely to be encountered. Surveys showed that large bMSCs contain 20–30 messages, while most bMSCs contain between five and fifteen messages. Thus, the approach can be seen as applicable to genuine specifications. Therefore, we will not take the time needed for model creation into account during the experimental setup defined in Sect. 5. We consider the time acceptable since the manual review is much more time-consuming than the review model generation.

5 Empirical evaluation

We evaluated the dedicated review model in close collaboration with our industrial partners from the automotive and the avionics domain, including original equipment manufacturers as well as suppliers (cf. [9]). In this evaluation we used a real-world specification of an automotive adaptive exterior lighting system. The evaluation indicated the usefulness of the dedicated review model in industry.

To get insights whether the use of the dedicated review models can significantly increase the effectiveness and efficiency of manual reviews, we designed three controlled experiments. The experiments compare manual reviews of the behavioral requirements and the functional design against stakeholder intensions using the dedicated review model with manual reviews without the dedicated review model. Our current validation does not take the evolution of behavioral requirements and functional design into account and thus does not consider increments.

5.1 Research questions

We defined three research questions (RQ1–RQ3) to investigate whether the proposed dedicated review model is beneficial for the reviews of behavioral requirements and functional design against the stakeholder intensions. We therefore compare reviews using the dedicated review model with reviews not using the dedicated review model. In addition, we evaluated whether the dedicated review model has different effects on users with different experiences. In detail, we defined the following research questions:

- *RQ1: Does the use of a dedicated review model have an impact on the effectiveness and efficiency of the manual review of behavioral requirements and functional design?*
- *RQ2: Does the experience and skill level of reviewers have an influence on the effectiveness and efficiency of reviews using dedicated review models?*
- *RQ3: Does the chosen notation format of a dedicated review model have an impact on the effectiveness and efficiency of the review?*

5.2 Experimental setup

5.2.1 Overview

To investigate the research questions defined in Sect. 5.1 we conducted three experiments. The experimental setups and early results of the experiments have been reported in [38]. In the experiments, the manual review of the dedicated review model is compared to the manual review of the original artifacts (i.e., behavioral requirements and functional design). The manual reviews are compared w.r.t. effectiveness, efficiency, user confidence and subjective supportiveness in order to contribute to research question RQ1. Furthermore, experiments 1 and 2 are conducted with participants with different experience and skill level. Hence, a comparative analysis between the results of experiments 1 and 2 can contribute to RQ2. In addition, experiment 3 compared the review of the review model with the review of the functional design only. Thereby, effects that might stem from the size and complexity of reviewing two artifacts at one time are prevented. Thus, statements regarding the impact of the notation format as outlined by RQ3 can be made. Table 1 gives an overview on the three experiments reported in this article and summarizes their major differences in the experimental setup. The dependent and independent variables as well as the hypothesis of the experiments will be detailed in the subsequent subsections.

5.2.2 Participants

The experiments were either conducted with graduate students or with undergraduate students as participants. We decided to use student participants for the following reasons:

- We aim at ensuring high conclusion validity as this is important for hypothesis testing experiments [39]. The use of student participants, thus, allowed us to conduct controlled experiments to strengthen conclusion validity.
- The use of student participants has been valued as fair comparison (see [40, 41]). Otherwise, there is a risk that participants (i.e., industry professionals) have more profound knowledge and are more trained with the control technique compared to the treatment technique (i.e., the approach under investigation).
- Although the use of student participants must be regarded critical (e.g., [42]), existing empirical research shows that findings gained in experiments with student participants are generalizable to industry practitioners under specific circumstances (e.g., [43–45]).
- We also conducted pretests with industry professionals, but the low number of participants prevented reaching statistical significance. However, the pretest results did not differ from the experimental results obtained with the students.

Table 2 Overview of the participants in the experiments

Experiment	Participants	*N* (number of participants)
Experiment 1	Graduate students	21
Experiment 2	Undergraduate students	125
Experiment 3	Graduate students	41

Nevertheless, experiments with students introduce a threat to external validity, especially the question whether the findings obtained also hold in industrial practice.

Experiments 1 and 3 used graduate students as experiment participants, while experiment 2 used undergraduate students. Graduate and undergraduate students were recruited by mandatory exercises within university courses, as the experiments contributed to the courses teaching goals. No bonuses with regard to the courses final examination were given to avoid social threats to validity. In so far, the recruitment strategy has to be seen as *opportunity sampling*. We used a *within-subject design*, all participants acted as treatment and control. The order in which the participants were assigned to treatment and to control exercises was randomized. Subsequently, we will briefly outline the course-specific aspects of the recruited student participants:

- *Graduate students* Student participants for experiments 1 and 3 were recruited within master-level university courses on requirements engineering. The experiments were conducted after lessons on validation techniques for requirements. The participants are mainly holding bachelor degrees in "Systems Engineering" (with particular emphasis on software engineering) or "Business Information Systems." In general, these students are also enrolled in master-level degree programs of "Systems Engineering" or "Business Information Systems."
- *Undergraduate students* For conducting experiment 2, we replicated experiment 1 in a bachelor-level requirements engineering course. In contrast to the master-level course, the experiment was conducted as an introduction to requirements validation. Hence, it was ensured that, besides the difference of graduate and undergraduate students, experience and skill level between participants in experiments 1 and 2 was remote, since experiment 1 participants were introduced to validation techniques, while experiment 2 participants used the experiment as introduction. The participants are mainly enrolled in bachelor-level degree programs for "Systems Engineering," "Busi-

Table 3 Overview of review styles compared in the experiments

Experiment	Review Style 1	Review Style 2
Experiment 1	Review Style SP	Review Style RM
Experiment 2	Review Style SP	Review Style RM
Experiment 3	Review Style FD	Review Style RM

ness Information Systems" or less often in "Business Administration."

Table 2 gives an overview on the groups of participants as well as the number of participants in each of the three experiments.

Note that experiments 1 and 3 have also been replicated, but as results do not significantly differ, we continue reporting the results of the original experiments.

5.2.3 Experimental material

The participants had to conduct a review of specification artifacts characterizing a collision avoidance system for aircraft (for experiments 1 and 2). We used an industrial sample specification from the avionics domain as source for the experimental material. Additionally, the experimental material contains stakeholder intentions for the aircraft collision avoidance system. To ensure the generalizability to a real industrial engineering process, we prepared the experimental material in close collaboration with industry professionals from a large European company in the avionics domain. Furthermore, industry professionals from the automotive domain were involved in discussions about the experimental material to allow for generalizability not only to the avionics domain.

For experiment 3, we used an industrial sample specification of a lane-keeping support system from the automotive domain. This experimental material was developed in collaboration with experts from a large German car maker and a large German automotive first-tier supplier. Also experts from the avionics domain were involved in the discussions.

The experimental material (i.e., excerpts from the sample specifications) and the layout of the generated review models have been manually adjusted to fit screen size and to improve readability.

5.2.4 Independent variables

As independent variable within the conducted experiments, we investigated different *review styles* to validate the specified behavior against the actual stakeholder intentions:

- *Review Style RM (short: RM)* The participants use the dedicated review model, which integrates the information specified in the behavioral requirements and the functional design in one review artifact.
- *Review Style SP (short: SP)* The participants use the original specifications of behavioral requirements and functional design as review artifacts.
- *Review Style FD (short: FD)* The participants use the original specification of the functional design as review artifact.

Table 3 shows the review styles used in each experiment. While in experiments 1 and 2 SP is compared to RM, in experiment 3 the review styles FD and RM are compared. In the latter case, of course, only information contained within the functional design is under investigation.

5.2.5 Dependent variables

The dependent variables do not differ between the three experiments. As dependent variables we defined effectiveness, efficiency, user confidence and subjective supportiveness. Subsequently, we briefly outline the defined meaning of each dependent variable:

- *Effectiveness* The ratio of correctly identified and correctly rejected stakeholder intentions.
- *Efficiency* The average time spent for one correctly identified or correctly rejected stakeholder intention.
- *User confidence* The average confidence a participant claims for identifying and rejecting a stakeholder intention.
- *Subjective supportiveness* Average result of self-rated standardized questionnaire items from the TAM 3 (Technology Acceptance Model v.3, cf. [46]) to rate RM against SP for perceived usefulness, perceived ease of use and computer self-efficacy.

Additionally, we measured several *covariates* such as highest educational achievement, degree program, semester, age, gender, as well as participants' self-rated experience in six categories related to conducting reviews in general and related to the used modeling notations in particular.

5.2.6 Experimental design

The study's experimental setup consists of a controlled experiment (to determine effectiveness, efficiency and user confidence) and a post hoc questionnaire (to determine subjective supportiveness and covariates). The experiment uses a within-subject design. Each participant conducts a review of the specifications of an avionic collision avoidance system in both review styles (*SP* and *RM* for experiments 1 and

2; *FD* and *RM* for experiment 3). The order of the review style was randomized for each participant. Each participant reviews the models against 24 stakeholder intentions, 12 in *SP* or *FD* and 12 in *RM*. The stakeholder intentions, which are represented in the artifacts, were identical in *SP/FD* and *RM*.

For experiments 1 and 2, in this setup "reviewing" means deciding for each stakeholder intention which of the following cases applies: (1) The intention was correctly reflected in the behavioral requirements only; (2) the intention was correctly reflected in the functional design only; (3) the intention was correctly reflected in the behavioral requirements and the functional design; (4) the intention was correctly reflected neither in the functional design nor in the behavioral requirements. For experiment 3 "reviewing" means deciding whether a stakeholder intention is correctly reflected in the functional design or not.

In addition, in all experiments the participants were asked to rate their confidence in each decision on a 5-point Likert scale. After the review, each participant was asked for some personal data and his or her attitude toward the supportiveness of the different review styles.

The controlled experiments were conducted as online experiments. We used an online questionnaire to minimize interaction effects between participants. The experiments are designed to last about 20 min. This is due to minimize participants' mortality because of losing interest. Students typically participated from home. The online questionnaires can be accessed at: https://doi.org/10.5281/zenodo.1744665.

5.3 Hypotheses

Based on the independent and dependent variables as well as the defined experiments and the comparison between them, hypotheses for each research question are defined.

5.3.1 Hypotheses for research question 1

The null and alternative hypotheses for research question RQ1 are related to the comparison of the review styles *SP* and *RM*. The principal hypotheses are:

- $\mathbf{H_{eff}1\text{-}0}$**:** There is no significant difference between the effectiveness of *SP* and *RM*.
  $\mathbf{H_{eff}1\text{-}a}$**:** *RM* is significantly more effective than *SP*.
- $\mathbf{H_{efy}1\text{-}0}$**:** There is no significant difference between the efficiency of *SP* and *RM*.
  $\mathbf{H_{efy}1\text{-}a}$**:** *RM* is significantly more efficient than *SP*.
- $\mathbf{H_{uco}1\text{-}0}$**:** There is no significant difference between the participants' confidence in *SP* and *RM*.
  $\mathbf{H_{uco}1\text{-}a}$**:** Participants are significantly more confident in *RM* than in *SP*.
- $\mathbf{H_{sup}1\text{-}0}$**:** There is no significant difference between the subjective supportiveness of *SP* and *RM*.
  $\mathbf{H_{sup}1\text{-}a}$**:** *RM* is significantly more supportive than *SP*.

Considering the distinction between the set of participants in experiment 1 and experiment 2, each of these hypotheses can be refined into a more fine-grained hypothesis related to the group of participants under investigation. Table 4 shows the detailed list of hypotheses derived for research question RQ1.

5.3.2 Hypotheses for research question 2

While research question RQ1 deals with the comparison of effectiveness, efficiency, user confidence and subject supportiveness for the review styles RM and SP, research question RQ2 deals with the impact of the reviewers' experiences and skills. This is important as reviews are commonly conducted involving the stakeholders. Hence, the question arises whether stakeholders, which are more unexperienced with the modeling language or reviews in general, benefit the use of a dedicated review model. Therefore, we compare the results of experiments 1 and 2 w.r.t. the influence of the experience the participants have (i.e., experienced graduate students compared to unexperienced undergraduate students). Table 5 defines the detailed hypotheses for research question RQ2.

5.3.3 Hypotheses for research question 3

Similar to the principal hypotheses from Sect. 5.3.2 the hypotheses for research question RQ3 are defined. Here, the focus is on comparing the review styles *FD* and *RM*. The hypotheses are shown in Table 6.

6 Evaluation results

In this section, we report the descriptive statistics, the hypotheses tests, as well as the threats to validity of the three experiments.

Regarding the data set preparation, we filtered some of the participants' data sets from the final data set. This was necessary in the bachelor course (experiment 2), since some of the participants obviously did not perform serious reviews. These participants finished one reviewing task (consisting of the validation of 12 natural language stakeholder intentions and the rating of participants' confidence for each) in less than 1 min. In total, we used 119 data sets for analysis in the bachelor course. In the master courses (experiments 1 and 3), it was not necessary to filter participants. We used SPSS

Table 4 Hypotheses for research question RQ1

Independent variable	Experiment	Hypothesis	Description
Effectiveness	Experiments 1 and 2	$\mathbf{H_{eff}1\text{-}0}$	There is no significant difference between the effectiveness of *SP* and *RM*
		$\mathbf{H_{eff}1\text{-}a}$	*RM* is significantly more effective than *SP*
	Experiment 1	$\mathbf{H_{eff}1.1\text{-}0}$	There is no significant difference between the effectiveness of graduates in *SP* and *RM*
		$\mathbf{H_{eff}1.1\text{-}a}$	*RM* is significantly more effective than *SP* for graduates
	Experiment 2	$\mathbf{H_{eff}1.2\text{-}0}$	There is no significant difference between the effectiveness of undergraduates in *SP* and *RM*
		$\mathbf{H_{eff}1.2\text{-}a}$	*RM* is significantly more effective than *SP* for undergraduates
Efficiency	Experiments 1 and 2	$\mathbf{H_{efy}1\text{-}0}$	There is no significant difference between the efficiency of *SP* and *RM*
		$\mathbf{H_{efy}1\text{-}a}$	*RM* is significantly more efficient than *SP*
	Experiment 1	$\mathbf{H_{efy}1.1\text{-}0}$	There is no significant difference between the efficiency of graduates in *SP* and *RM*
		$\mathbf{H_{efy}1.1\text{-}a}$	*RM* is significantly more efficient than *SP* for graduates
	Experiment 2	$\mathbf{H_{efy}1.2\text{-}0}$	There is no significant difference between the efficiency of undergraduates in *SP* and *RM*
		$\mathbf{H_{efy}1.2\text{-}a}$	*RM* is significantly more efficient than *SP* for undergraduates.
User confidence	Experiments 1 and 2	$\mathbf{H_{uco}1\text{-}0}$	There is no significant difference between the participants' confidence in *SP* and *RM*
		$\mathbf{H_{uco}1\text{-}a}$	Participants are significantly more confident in *RM* than in *SP*
	Experiment 1	$\mathbf{H_{uco}1.1\text{-}0}$	There is no significant difference between the graduate participants' confidence in *SP* and *RM*
		$\mathbf{H_{uco}1.1\text{-}a}$	Graduate participants are significantly more confident in *RM* than in *SP*
	Experiment 2	$\mathbf{H_{uco}1.2\text{-}0}$	There is no significant difference between the undergraduate participants' confidence in *SP* and *RM*
		$\mathbf{H_{uco}1.2\text{-}a}$	Undergraduate participants are significantly more confident in *RM* than in *SP*
Subjective supportiveness	Experiments 1 and 2	$\mathbf{H_{sup}1\text{-}0}$	There is no significant difference between the subjective supportiveness in *SP* and *RM*
		$\mathbf{H_{sup}1\text{-}a}$	*RM* is significantly more supportive than *SP*
	Experiment 1	$\mathbf{H_{sup}1.1\text{-}0}$	There is no significant difference between the subjective supportiveness in *SP* and *RM* for graduates
		$\mathbf{H_{sup}1.1\text{-}a}$	*RM* is significantly more supportive than *SP* for graduates
	Experiment 2	$\mathbf{H_{sup}1.2\text{-}0}$	There is no significant difference between the subjective supportiveness in *SP* and *RM* for undergraduates
		$\mathbf{H_{sup}1.2\text{-}a}$	*RM* is significantly more supportive than *SP* for undergraduates

to analyze the data sets. Therefore, there was no need to complement data sets with missing measurements (cf. [47]).

6.1 Descriptive statistics

6.1.1 Results of experiments 1 and 2

Table 7 shows mean, median, standard deviation minimum and maximum of the dependent variables effectiveness, efficiency, confidence and subjective supportiveness separately for *RM* and *SP* as well as graduate (G) and undergraduates (U). Effectiveness represents the percentage of correct review decisions. Efficiency is measured in minutes. Confidence and subjective supportiveness are both measured on a 5-point Likert scale, 1 indicating very unconfident and 5 indicating very confident, and 1 indicating that RM is found to be most supportive and 5 indicating that SP is found to be most supportive, respectively.

As both, the descriptive statistics as well as the boxplots in Fig. 16 clearly indicate that *RM* is more effective, more efficient and more confidence inducing (compared with *SP)*. Also, graduate students are on average more effective, efficient and confident than undergraduates, even though they expend an almost equal amount of effort.

Participants were asked to rate which review style (i.e., *RM* or *SP*) was more supportive. This was carried out by answering ten questions from TAM3 to rate the perceived usefulness, perceived ease of use and computer self-efficacy. Since the TAM3 only intends to rate one technique on a Likert scale, and we used a 5-point Likert scale to rate both techniques against each other, there is a risk that the reliability of the measurements could be corrupted. We calculated

Table 5 Hypotheses for research question RQ2

Independent variable	Experiment	Hypothesis	Description
Effectiveness	Comparison Exp. 1 and 2	$\mathbf{H_{eff}2.1\text{-}0}$	There is no significant difference between the effectiveness of undergraduates and graduates in *SP*
		$\mathbf{H_{eff}2.1\text{-}a}$	Graduates are more effective than undergraduates in *SP*
	Comparison Exp. 1 and 2	$\mathbf{H_{eff}2.2\text{-}0}$	There is no significant difference between the effectiveness of undergraduates and graduates in *RM*
		$\mathbf{H_{eff}2.2\text{-}a}$	Graduates are more effective than undergraduates in *RM*
	Comparison Exp. 1 and 2	$\mathbf{H_{eff}2.3\text{-}0}$	There is no significant difference between graduates and undergraduates regarding their distance in effectiveness between *SP* and *RM*
		$\mathbf{H_{eff}2.3\text{-}a}$	The distance in effectiveness between *SP* and *RM* is larger for graduates than for undergraduates
Efficiency	Comparison Exp. 1 and 2	$\mathbf{H_{efy}2.1\text{-}0}$	There is no significant difference between the efficiency of undergraduates and graduates in *SP*
		$\mathbf{H_{efy}2.1\text{-}a}$	Graduates are more efficiency than undergraduates in *SP*
	Comparison Exp. 1 and 2	$\mathbf{H_{efy}2.2\text{-}0}$	There is no significant difference between the efficiency of undergraduates and graduates in *RM*
		$\mathbf{H_{efy}2.2\text{-}a}$	Graduates are more efficiency than undergraduates in *RM*
	Comparison Exp. 1 and 2	$\mathbf{H_{efy}2.3\text{-}0}$	There is no significant difference between graduates and undergraduates regarding their distance in efficiency between *SP* and *RM*
		$\mathbf{H_{efy}2.3\text{-}a}$	The distance in efficiency between *SP* and *RM* is larger for graduates than for undergraduates
User confidence	Comparison Exp. 1 and 2	$\mathbf{H_{uco}2.1\text{-}0}$	There is no significant difference between the confidence of undergraduates and graduates in *SP*
		$\mathbf{H_{uco}2.1\text{-}a}$	Graduates are more confident than undergraduates in *SP*
	Comparison Exp. 1 and 2	$\mathbf{H_{uco}2.2\text{-}0}$	There is no significant difference between the confidence of undergraduates and graduates in *RM*
		$\mathbf{H_{uco}2.2\text{-}a}$	Graduates are more confident than undergraduates in *RM*
	Comparison Exp. 1 and 2	$\mathbf{H_{uco}2.3\text{-}0}$	There is no significant difference between graduates and undergraduates regarding their distance in confidence between *SP* and *RM*
		$\mathbf{H_{uco}2.3\text{-}a}$	The distance in confidence between *SP* and *RM* is larger for graduates than for undergraduates
Subjective supportiveness	Comparison Exp. 1 and 2	$\mathbf{H_{sup}2\text{-}0}$	There is no significant difference between the subjective supportiveness for undergraduates and graduates
		$\mathbf{H_{sup}2\text{-}a}$	Graduates are more supported than undergraduates

Table 6 Hypotheses for research question RQ3

Independent variable	Experiment	Hypothesis	Description
Effectiveness	Experiment 3	$\mathbf{H_{eff}3\text{-}0}$	There is no significant difference between the effectiveness of *FD* and *RM*
		$\mathbf{H_{eff}3\text{-}a}$	*RM* is significantly more effective than *FD*
Efficiency	Experiment 3	$\mathbf{H_{efy}3\text{-}0}$	There is no significant difference between the efficiency of *FD* and *RM*
		$\mathbf{H_{efy}3\text{-}a}$	*RM* is significantly more efficient than *FD*
User confidence	Experiment 3	$\mathbf{H_{uco}3\text{-}0}$	There is no significant difference between the participants' confidence in *FD* and *RM*
		$\mathbf{H_{uco}3\text{-}a}$	Participants are significantly more confident in *RM* than in *FD*
Subjective supportiveness	Experiment 3	$\mathbf{H_{sup}3\text{-}0}$	There is no significant difference between the subjective supportiveness in *FD* and *RM*
		$\mathbf{H_{sup}3\text{-}a}$	*RM* is significantly more supportive than *FD*

Cronbach's alpha to determine the reliability of our measurements. In the undergraduate experiment the resulting Cronbach's α is as follows: for *perceived usefulness*: $\alpha(5) = 0.919$; for *perceived ease of use*: $\alpha(4) = 0.919$; for *computer self-efficacy*: $\alpha(4) = 0.719$; and for the determination of the overall *supportiveness*: $\alpha(4) = 0.921$. In consequence, we are confident to say that our adopted measurements are reliable.

Table 7 Descriptive statistics for effectiveness, efficiency, confidence and subjective supportiveness of RM and SP

		Mean	95% confidence interval for mean		5% Trimmed means	Median	Variance	SD	Min	Max	Mean std. error
			Lower bound	Upper bound							
Effect. RM	U	.553419	.515063	.591774	.554725	.583333	.043877	.209469	.083333	1.000000	.019365
	G	.730159	.618334	.841983	.741623	.833333	.060351	.245663	.250000	1.000000	.053608
Effect. SP	U	.396011	.367612	.424411	.392134	.333333	.024055	.155096	.083333	.833333	.014339
	G	.484127	.423412	.544842	.482584	.500000	.017791	.133383	.250000	.750000	.029107
Effcy. RM	U	1.874129	1.576917	2.171340	1.650319	1.566667	2.634590	1.623142	.129167	10.900000	.150060
	G	1.420066	.994644	1.845488	1.345306	1.060417	.873465	.934594	.354545	3.883333	.203945
Effcy. SP	U	2.764499	2.452134	3.076864	2.632740	2.529167	2.910089	1.705898	.241667	10.875000	.157710
	G	2.288454	1.794752	2.782157	2.264400	2.186667	1.176350	1.084597	.229167	4.744444	.236678
Conf. RM	U	3.706553	3.554798	3.858307	3.758784	3.750000	.686853	.828766	1.000000	5.000000	.076619
	G	4.313492	3.950463	4.676521	4.402337	4.500000	.636045	.797524	2.000000	5.000000	.174034
Conf. SP	U	3.475783	3.337955	3.613612	3.523821	3.500000	.566578	.752714	1.000000	4.750000	.069588
	G	3.833333	3.381152	4.285515	3.914683	4.000000	.986806	.993381	1.166667	5.000000	.216773
Subject. support.	U	2.495489	2.346216	2.644762	2.484396	2.361111	.664577	.815216	1.000000	4.666667	.075367
	G	2.056878	1.753587	2.360169	2.043357	2.000000	.443941	.666289	1.000000	3.361111	.145396

6.1.2 Results of experiment 3

Table 8 shows the descriptive statistics for effectiveness, review effort, efficiency confidence and subjective supportiveness. The values were measured as described in Sect. 6.1.1.

The descriptive statistics as well as the boxplots in Fig. 17 show that *RM* is more effective, efficient and inducing a higher confidence than *FD*. Furthermore, *RM* is seen as more supportive than *FD*.

As in the previous experiments, the reliability of the questionnaire for determining the subjective supportiveness must be examined. The questionnaire is based on the standardized TAM3 questions as well, but here the questions are not used to evaluate a single instrument but to compare two instruments (review styles *RM* and *SP*). Thus, the reliability of this measurement has to be evaluated in the changed setting. The results of calculating Cronbach's alpha show that the reliability of *perceived ease of use*, *perceived usefulness* and *subjective supportiveness* is excellent ($\alpha > 0.9$), and the reliability of *computer self-efficacy* is good ($\alpha > 0.8$).

6.2 Hypotheses tests

All hypotheses tests rely on the *t* test. For hypotheses regarding effectiveness, efficiency and user conference two-tailed repeated-measures t tests are used. For hypotheses regarding subjective supportiveness two-tailed one-sample t tests are used. Tests for preconditions to ensure feasibility of t tests were conducted, and results are in correspondence with commonly seen appropriate conditions, particularly under consideration of large sample sizes, as outlined in [48–50].

6.2.1 Research question 1

6.2.1.1 Effectiveness On average, the undergraduate students made 54% correct decisions when using the review model (i.e., review style *RM*) and 39% when using the original specifications (i.e., *SP*). The graduate students made 73% correct decisions when using the review model and 48% when using the original specifications. Regarding significance, in both cases a two-tailed repeated-measures t test (see Table 9) indicates very high significance.

In addition, a performed post hoc power analysis (cf. [51]) indicates that the number of participants was sufficient to claim significance in both experiments. For undergraduate students, the effect size is 0.715 and power is 1.0 (for $p < 0.01$). For graduate students the effect size is 0.89 and power is 0.94 (for $p < 0.01$). We can conclude that there is a high significance between the differences in effectiveness of *SP* and *RM*. *RM* is in both cases significantly more effective than *SP*. We can thus **reject H_{eff}1.1-0** and **H_{eff}1.2-0** and

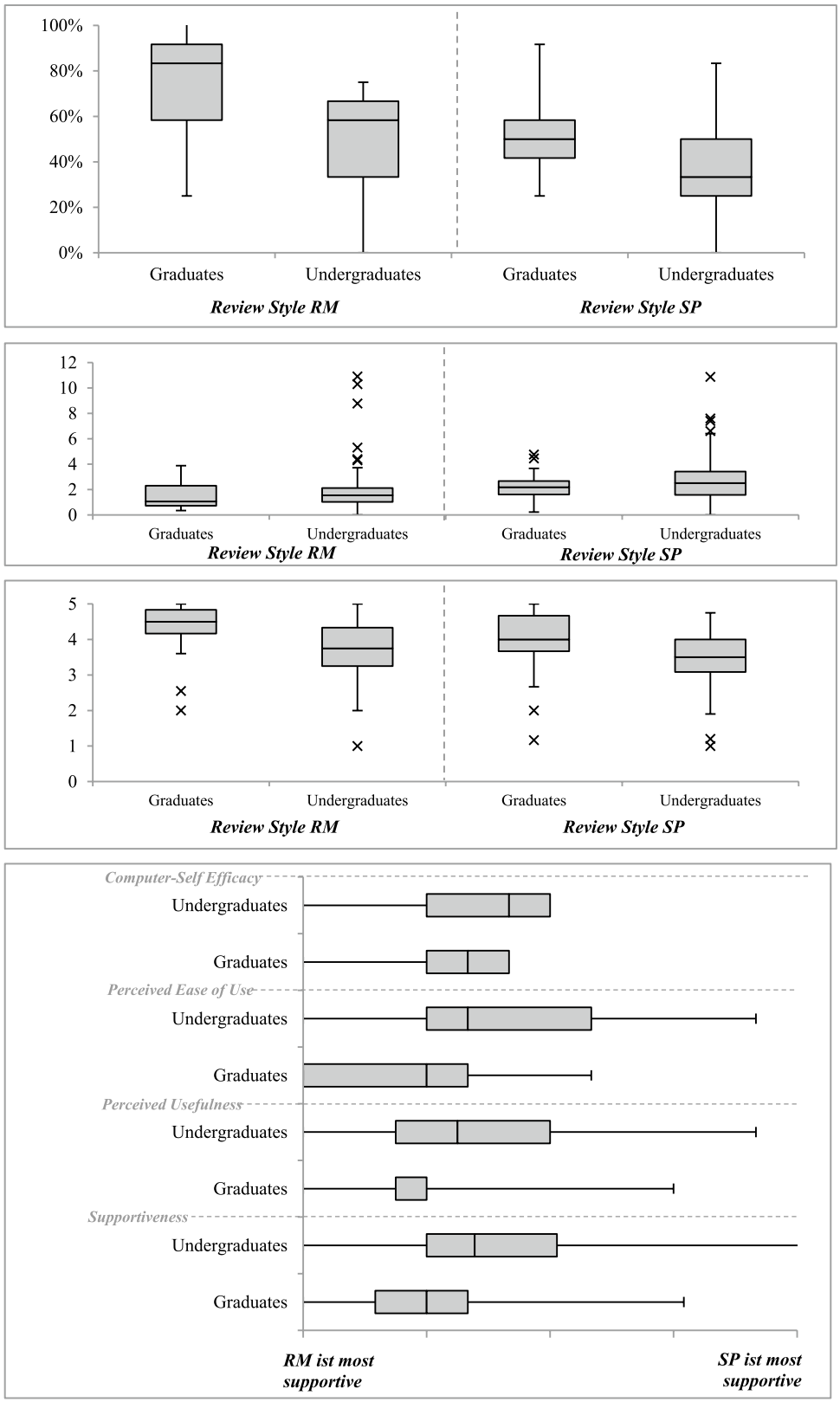


Fig. 16 Boxplots for effectiveness, efficiency, confidence and subjective supportiveness for comparison of RM and SP for graduates and undergraduates

Table 8 Descriptive statistics for effectiveness, efficiency, confidence and subjective supportiveness of RM and SP

	Mean	95% confidence interval for mean		5% Trimmed means	Median	Variance	SD	Min	Max	Mean std. error
		Lower bound	Upper bound							
Effect. FD	.773519	.738993	.808045	.778068	.785714	.011965	.109384	.500000	.928571	.017083
Effect. RM	.890244	.856766	.923722	.901665	.928571	.011249	.106063	.428571	1.000000	.016564
Effcy. FD	1.267365	.939659	1.595072	1.124910	1.092857	1.077928	1.038233	.138889	6.175000	.162145
Effcy. RM	1.054110	.817494	1.290726	.990894	.790278	.561963	.749642	.176923	3.329167	.117074
Conf. FD	3.848463	3.665224	4.031703	3.872829	3.929000	.337022	.580536	2.286000	4.786000	.090665
Conf. RM	4.231634	4.067891	4.395377	4.255157	4.286000	.269119	.518766	3.000000	5.000000	.081018
Support.	1.987805	1.731980	2.243630	1.902590	2.000000	.656908	.810498	1.000000	4.666667	.126579

accept $\mathbf{H_{eff}1.1\text{-}a}$ and **$\mathbf{H_{eff}1.2\text{-}a}$**. Hence, we can also **reject** the overall null-hypothesis **$\mathbf{H_{eff}1\text{-}0}$** and **accept $\mathbf{H_{eff}1\text{-}a}$**.

6.2.1.2 Efficiency Regarding efficiency, we determined the ratio of time usage per correct answer. For undergraduates *RM* is 57.5% more efficient than *SP*. For graduates *RM* is 61% more efficient. Graduates are 29.8% more efficient in *RM* than undergraduates are and 18.8% more efficient in *SP*.

Results of the t test (see Table 10, mean values in seconds) indicate that *RM* is highly significantly more efficient than *SP* in both experiments. Power analysis shows effect sizes of 0.37 (undergraduates) and 0.81 (graduates). Power is 0.92 (undergraduates, $p < 0.01$) and 0.79 (graduates, $p < 0.01$), which we consider to be adequate. Hence, we can **reject $\mathbf{H_{efy}1.1\text{-}0}$** and **$\mathbf{H_{efy}1.2\text{-}0}$**, and we can **accept $\mathbf{H_{efy}1.1\text{-}a}$** and **$\mathbf{H_{efy}1.2\text{-}a}$**. *RM* is highly significantly more effective than *SP* (**$\mathbf{H_{efy}1\text{-}a}$**).

6.2.1.3 User confidence In comparison with means, the user confidence of graduate students in *RM* was 12.5% higher than in *SP*. The confidence of undergraduate students in *RM* was 6.6% higher than their confidence in *SP*. Graduates' confidence in *RM* was 13.8% higher than undergraduates' confidence, and graduates confidence in *SP* was 10% higher than undergraduates' confidence in *RM*. In addition, Pearson's correlation ($\varrho(140) = 0.746$, $p < 0.001$) shows that participants with a high confidence in *RM* tend to have a high confidence in *SP* and vice versa.

Based on the t test results from Table 11 and the results of a power analysis, we can **reject $\mathbf{H_{uco}1\text{-}0}$** (as well as **$\mathbf{H_{uco}1.1\text{-}0}$** and **$\mathbf{H_{uco}1.2\text{-}0}$**) and **accept $\mathbf{H_{uco}1\text{-}a}$** (as well as **$\mathbf{H_{uco}1.1\text{-}a}$** and **$\mathbf{H_{uco}1.2\text{-}a}$**). User confidence is significantly higher for *RM* than for *SP*. Power analysis shows effect sizes of 0.43 (undergraduates) and 0.67 (graduates). Power is 0.98 (undergraduates, $p < 0.01$) and 0.6 (graduates, $p < 0.01$), which we consider to be acceptable.

6.2.1.4 Subjective supportiveness For subjective supportiveness t tests show significance in both experiments as well (see Table 12); also, power analysis is in accordance. Thus, we can **reject $\mathbf{H_{sup}1.1\text{-}0}$**, **$\mathbf{H_{sup}1.2\text{-}0}$** and consequently **$\mathbf{H_{sup}1\text{-}0}$**. We **accept $\mathbf{H_{sup}1.1\text{-}a}$**, **$\mathbf{H_{sup}1.2\text{-}a}$** and consequently **$\mathbf{H_{sup}1\text{-}a}$**. Regarding all measurements of supportiveness *RM* is rated significantly higher than *SP* in both experiments.

6.2.1.5 Meaning for the research question Since all null hypotheses are rejected, we can accept all alternative hypotheses. We thus have confidence in claiming that the proposed dedicated review model has a significant positive impact on the review of behavioral requirements and functional design against stakeholder intensions. The use of the review models is beneficial with respect to effectiveness and

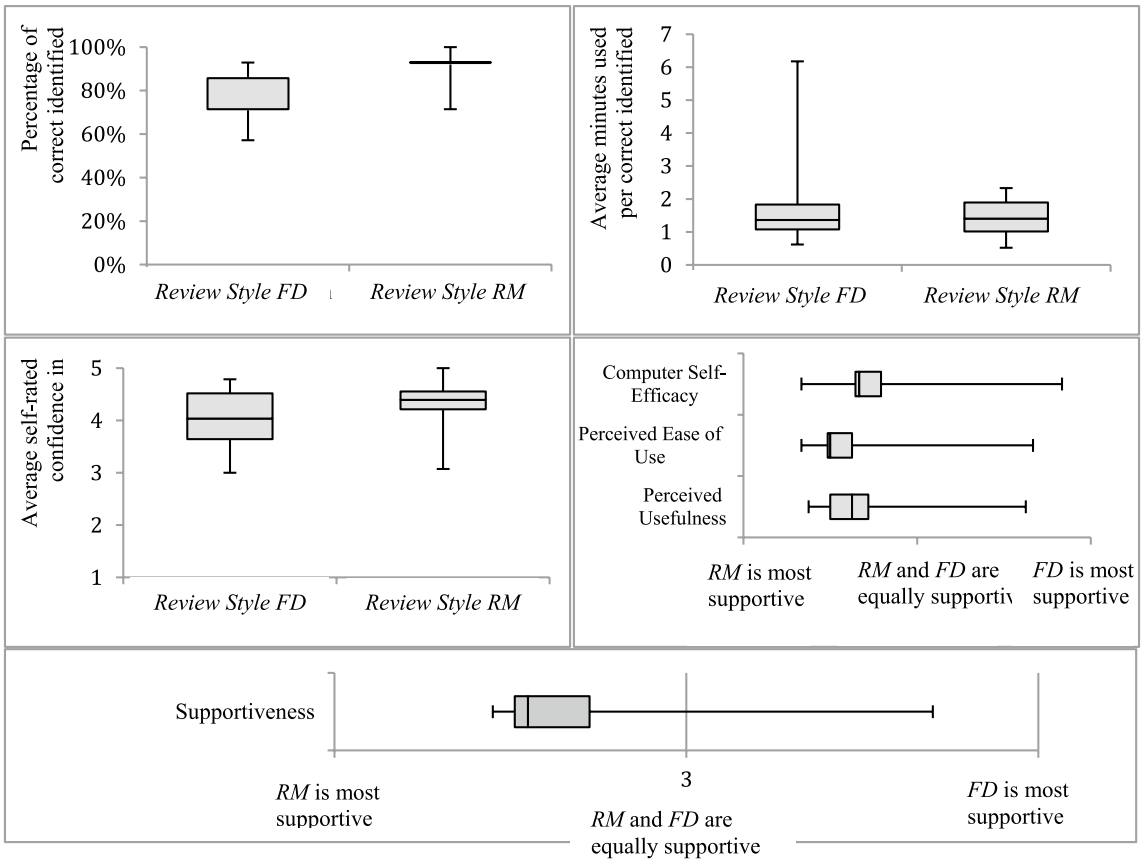


Fig. 17 Boxplots for effectiveness, efficiency, confidence and subjective supportiveness

Table 9 T test results for effectiveness

	RM		SP		Differences			
	Mean	SD	Mean	SD	Mean	SD	*t*	Sig
Graduates	.730	.246	.484	.133	.246	.248	4.549	.000
Undergraduates	.544	.220	.389	.162	.155	.217	7.797	.000

Table 10 T test results for efficiency

	RM		*SP*		Differences			
	Mean	SD	Mean	SD	Mean	SD	*t*	Sig
Graduates	1.420	.935	2.288	1.085	.868	1.073	− 3.71	.001
Undergraduates	1.843	1.627	2.718	1.729	0.875	2.372	− 4.03	.000

Table 11 T test results for user confidence

	RM		*SP*		Differences			
	Mean	SD	Mean	SD	Mean	SD	*t*	Sig
Graduates	4.313	.798	3.833	.993	.480	.787	2.797	.011
Undergraduates	3.709	.826	3.478	.751	.231	.541	4.662	.000

Table 12 T test results for subjective supportiveness

		Mean	SD	Diff. from expectation	*t*	Sig
PU	Graduates	1.964	.747	1.036	6.354	.000
	Undergraduates	2.435	.982	.565	6.277	.000
PEOU	Graduates	2.0	.919	1.0	4.987	.000
	Undergraduates	2.504	1.025	.496	5.277	.000
CSE	Graduates	2.206	.741	.794	4.905	.000
	Undergraduates	2.560	.888	.44	5.401	.000
Subjective supportiveness	Graduates	2.057	.666	.943	6.487	.000
	Undergraduates	2.5	.843	.500	6.476	.000

Table 13 Effectiveness in comparative analysis

	Graduates		Undergrad.		Differences			
	Mean	SD	Mean	SD	Mean	SD	*t*	Sig
RM	.73	.246	.544	.22	.186	.057	3.249	.003
SP	.484	.133	.389	.162	.095	.033	2.9	.007
Diff.	.246	.248	.155	.217	.091	.058	1.584	.125

Table 14 Efficiency in comparative analysis

	Graduates		Undergraduates		Differences			
	Mean	SD	Mean	SD	Mean	SD	*t*	Sig
RM	1.42	.935	1.843	1.627	.423	.253	1.672	.101
SP	2.288	1.085	2.718	1.729	.43	.285	1.508	.139
Diff.	.868	1.073	.875	2.372	.007	.528	.013	.989

Table 15 User confidence in comparative analysis

	Graduates		Undergraduates		Differences			
	Mean	SD	Mean	SD	Mean	SD	*t*	Sig
RM	4.313	.798	3.709	.826	.604	.19	3.183	.004
SP	3.833	.993	3.478	.751	.355	.227	1.561	.132
Diff.	.480	.787	.231	.541	.249	.179	1.394	.176

Table 16 Subjective supportiveness in comparative analysis

	Graduates		Undergraduates		Differences			
	Mean	SD	Mean	SD	Mean	SD	*t*	Sig
PU	1.964	.747	2.435	.982	.471	.225	2.089	.039
PEOU	2.000	.919	2.504	1.025	.504	.221	2.277	.030
CSE	2.206	.741	2.594	.769	.387	.176	2.195	.037
Supp.	2.057	.666	2.511	.82	.454	.164	2.774	.009

efficiency of the review, as well as to users' confidence in review decision making, and subjective supportiveness.

6.2.2 Research question 2

6.2.2.1 Effectiveness With respect to the difference between the review styles *RM* and *SP,* the results show that the correct identification of stakeholder intentions is consistently higher for graduate students than for undergraduate students. In addition, the mean of the differences is higher. Table 13 presents the t test results of a comparative analysis of effectiveness when using the review styles *RM* and *SP*, as well as the difference between *RM* and *SP.*

Table 17 T test for experiment 3

	RM		FD		Differences			
	Mean	SD	Mean	SD	Mean	SD	*t*	Sig
Effectiveness	.890	.106	.774	.109	.117	.135	5.524	.000
Efficiency	1.054	.75	1.267	1.038	.213	.955	1.430	.160
User confidence	4.232	.519	3.484	.581	.383	.441	5.558	.000

Thus, graduates are highly significantly more effective in both *SP* and *RM* than undergraduates. Power analysis shows effect sizes of .779 in *SP* with a power of .75 for $p < .01$ (or a power of .90 for $p < .05$) and .609 in *RM* with a power of .72 for $p < .05$. Hence, we **reject H_{eff}2.1-0** and **H_{eff}2.2-0,** and we **accept H_{eff}2.1-a** and **H_{eff}2.2-a**. While the difference between graduate and undergraduate students is higher for review style *RM* than for *SP*, no significance level is reached, i.e., we **cannot reject H_{eff}2.3-0**.

6.2.2.2 Efficiency Regarding the comparative analysis of efficiency between the results of experiments 1 and 2, it can be seen from Table 14 that, in both review styles, graduate students were in mean more efficient than undergraduate students. However, as the t tests do not show significance we **cannot reject H_{efy}2.1-0, H_{efy}2.2-0.** Regarding the differences between the review styles *RM* and *SP*, we can see that differences are almost identical for graduates and undergraduates. Hence, we **cannot reject H_{efy}2.3-0** either.

6.2.2.3 User confidence Table 15 shows the t test results for confidence. Graduates were significantly more confident in *RM* than undergraduates. Power analysis shows a medium effect size of .397 and a power of .98 for $p < 0.01$. Hence, we can **reject H_{uco}2.2-0** and **accept H_{uco}2.2-a**. As we cannot claim significance in *SP*, we **cannot reject H_{uco}2.1-0**. In comparison with the differences, it shows that the difference between user confidence in *RM* and *SP* is higher for graduates but not reaching significance. Hence, we **cannot reject H_{uco}2.3-0**.

6.2.2.4 Subjective supportiveness Comparing the supportiveness between bachelor and master participants, it is to note that the graduate students rated *RM* significantly more supportive in all measurements (i.e., *perceived usefulness*, *perceived ease of use* and *computer self-efficacy*). The results are depicted in Table 16 (subjective supportiveness was measured on a 5-point Likert scale, 1 indicating RM is found to be most supportive and 5 indicating SP is found to be most supportive). Power analysis for subjective supportiveness shows a medium effect size with a power of .338 for $p < .05$. Hence, we **reject H_{sup}2-0** and **accept H_{sup}2-a**.

6.2.2.5 Meaning for the research question As the results indicate, when using the dedicated review model, the experience and skill level of a reviewer has an influence on the effectiveness of the review. Hypotheses tests show that advanced users (i.e., graduate students compared to undergraduate students) are significantly more effective in conducting reviews in both review styles. However, the difference in effectiveness does not significantly increase, meaning that the review model supports experienced and inexperienced reviewers equally.

In addition, hypotheses tests regarding user confidence show that graduates and undergraduates are comparably confident in *SP*, while graduates are significantly more confident in *RM*. In other words, when using the review model, experienced reviewers are more supported in making confident decisions than inexperienced reviewers.

6.2.3 Research question 3

6.2.3.1 Effectiveness, efficiency and user confidence Table 17 summarizes the two-tailed repeated-measures t test results for *effectiveness*, *efficiency* and *user confidence*. While in mean 89% correct decisions have been made in RM, only 77.4% correct decisions have been made in FD. The results of the t test show that the difference is highly significant for $p < .001$. Power analysis shows a high effect size of .863, and power is .968 (for $p < .001$). Thus, we can **reject H_{eff}3-0** and **accept H_{eff}3-a**.

While there is also a difference in time usage per correct decision made (efficiency), t test results are not significant. Hence, we cannot reject **H_{efy}3-0**.

For user confidence, mean is 4.232 for RM and 3.484 for FD (measured on a Likert scale between 1 and 5). The t test results show that the difference is highly significant ($p < .001$). Power analysis results in a very high effect size

Table 18 T test results for subjective supportiveness

	Diff. from expectation	*t*	Sig
Perceived usefulness	.988	7.031	.000
Perceived ease of use	1.098	8.019	.000
Computer self-efficacy	.951	6.796	.000
Subjective supportiveness	1.012	7.997	.000

of 1.249 and a power of 1 (rounded after 3 decimals). We **reject H_{uco}3-0** and **accept H_{uco}3-a**.

6.2.3.2 Subjective supportiveness Differences for subjective supportiveness show highly significant differences for each measurement as well as for the combined value of subjective supportiveness (see Table 18). For subjective supportiveness power analysis results in a very high effect size of 1.249 and a power of 1 (rounded after 3 decimals). We **reject H_{sup}3-0** and **accept H_{sup}3-a**.

6.2.3.3 Meaning for the research question With the exception of efficiency, we could reject all null hypotheses and accept all defined alternative hypotheses. Thus, reviewers using the dedicated review model are more effective compared with reviewers using the functional design. Additionally, reviewers are more confident in using the review model. Moreover, they found the use of the dedicated review model more supportive than the functional design. Overall, the review model in its chosen notation positively impacts the effectiveness, the confidence and the subjective supportiveness.

7 Discussion

Section 7.1 briefly summarizes the major results of the reported experiments regarding the research questions. Section 7.2 discusses the threats to validity of the experiments, as well as mitigation strategies applied. Section 7.3 discusses the inferences that can be drawn to support reviews of model-based specifications.

7.1 Evaluation of results and implications

Hypotheses tests regarding the research questions (Sect. 5.1) show:

- The review of the review model is more effective, more efficient, more user confidence increasing and more subjective supportive than the review of the original specifications (i.e., behavioral requirements and functional design). (RQ1)
- The former holds true for experienced graduate students as well as for unexperienced undergraduate students. (RQ1)
- While graduate students are more effective than undergraduate students in reviews of the review model as well as of the original specifications, there is no effect discernable when comparing graduates and undergraduates regarding an increasing effectiveness due to review model use. (RQ2)
- Experienced and unexperienced participants are equally efficient and subjectively supported by the review model. Regarding confidence graduate participants are significantly more confident when using the review model than undergraduate participants. (RQ2)
- Comparing the review of the dedicated review model with the review of the functional design solely shows that the review of the review model is ranked higher in effectiveness, user confidence and subjective supportiveness. No effect regarding efficiency is recognizable. (RQ3)

In summary, these findings indicate that the proposed approach to use the dedicated review model to foster the review of behavioral requirements and functional design is beneficial, as the reviewers' effectiveness, efficiency and confidence are increased. Additionally, the approach is perceived more supportive than the review of the original specifications, or of the functional design. Furthermore, results show that the approach is equally suited to support experienced as well as unexperienced reviewers, which is of importance since stakeholders involved in a review must be considered of varying experience.

The results show that the use of a dedicated review model is beneficial for the review of behavioral requirements and the functional design against stakeholder intentions. Hence, the question arises whether review models can support manual validation in general, and which properties these review models should possess. As we are confident that positive effects of our review model stem from the merging of two specifications, as well as from the proposed notation format, we will elaborate on this question in more detail in Sect. 7.3 while taking the treats to validity from Sect. 7.2 into account.

7.2 Threats to validity

We need to discuss threats to validity on two different levels. First, we examine the threats to validity of the individual experiments, which we will discuss for all three experiments together due to their similar experimental setups. Second, we examine the threats to validity of the comparative analysis (i.e., threats affecting the comparability of experiment 1 and experiment 2) regarding research question RQ2.

7.2.1 Threats to validity of the individual experiments

To address threats to validity of the individual experiments, which exist for this type of study (cf. [39, 52]), we have employed certain strategies. The most relevant validity threats are discussed below:

Internal Validity We designed the experiment as an online questionnaire to be conducted within 20–30 min and gave the students a time frame of 5 days to participate. Thus,

we assume that internal threats to history, maturation or mortality do not apply. To avoid threats from compensatory equalization, we decided to use a within-subject design considering treatment and control in both experiments. The order of treatments was randomized among all students. In doing so, we avoid single group threats and reduce effects from interactions among subjects. Since volunteers may bias the results because they are generally more motivated than the whole course, we decided to conduct the experiment as mandatory part and explicitly decided to give no bonuses or credits as motivation. We avoided upfront briefing.

Construct Validity The fact that our experiments use mandatory student participation without student grading or other forms of bonuses aids in avoiding threats from evaluation apprehension. By using a minimum of teaching related to the experimental upfront and not conducting upfront briefing we also lower threats from hypothesis guessing by using naïve subjects. In addition, we only use quantitative measurements. In the post hoc questionnaire we adhere to standardized questions suggested by the TAM3.

The example specifications were carefully adapted in close collaboration with industry experts to fit the experimental setup. In addition, we used a pretest group to validate the experimental setup and material. The pretest group included professionals whose results were comparable to the results of graduate students. However, due to the small sample size of the pretest group, we did not achieve statistical significance. Pretest results were not included in the final results reported, as this would have led to heterogeneous groups impeding the comparability of graduate and undergraduate results.

With respect to efficiency, we must discuss some major threats to validity arising from the experimental setup. As we used an online questionnaire, which does not measure the time usage for a single decision but for the review of a whole diagram including confidence rating, we can make no statements about the time spent for reviewing each stakeholder intention. Since the experimental participation was done alone at home, we have no knowledge about time-consuming activities students might have conducted during their experimental participation. In the briefing, we stressed the need for focused work on the experiment. In addition, we designed the experiment in such a way that the experiment could be completed in less than 30 min to minimize the risk of losing focus. While we could not determine large irregularities, we cannot eliminate the issue that smaller activities (e.g., chatting or answering a phone call) could have influenced our measurement.

External Validity Our investigations showed that there are only minor differences between professional software developers and graduate software engineering students. However, the participation of undergraduate students is seen as threat to generalizability. Nevertheless, our findings indicate that there is no difference between the bachelor and master-level course with regard to the hypotheses, but that effect sizes are significantly higher in the graduate student experiment. In addition, we have carefully designed the example specifications in close collaboration with industry professional to ensure the representativeness of the experimental material for real-world settings.

Conclusion Validity For preparing our experiment in the example course, we used expert reviews of the experimental material in a first step, in order to avoid threats from unreliable measures. In addition, we used a pretest to validate the students' ability of understanding the material in the intended way. Note that the pretest participants were not chosen from the courses the participants were recruited in. Furthermore, we involved industry professionals to ensure that the transformation of industrial examples into experimental material did not corrupt the original intention of industrial problems and examples.

7.2.2 Threats to validity of the comparative analysis

The comparability of both experiments could be harmed if the same participants would participate in both experiments (i.e., first as a bachelor student and afterward as a master student). To avoid such effects, we conducted the experiment with graduate students first and the undergraduate student experiment afterward. In fact, we verified that there has not been double participation.

Since we want to generalize from undergraduate and graduate students to the level of skills and experience, we need to assure that the graduate students can be considered as more skilled and more experienced. For example, undergraduate students could also have work experience or already hold another degree. Therefore, we monitored the covariates and can state that to our knowledge the graduate students are more skilled and experienced than the undergraduates. Additionally, the experiment was conducted at different stages throughout the course setup, which also affects the skills and experiences regarding review techniques.

In the undergraduate experiment, also students of the degree program "business administration" were participating. However, no students of that program participated in the graduate experiment. By comparing the results among the different degree programs, we found no significant difference in the performance depending on the degree program in which a student is enrolled.

7.3 Inferences

As outlined in Sect. 7.1 and under consideration of the threats to validity in Sect. 7.2, the use of the proposed dedicated review model supports the review of behavioral requirements and functional design. This is in accordance

with the feedback obtained from professionals regarding the appropriateness and applicability of the dedicated review model. Furthermore, the results of the three experiments, supported by various discussions with engineers from industry, indicate that different properties of the review model have different effects.

The results of the experiments indicate that the notation format has an impact on the effectiveness, user confidence and subjective supportiveness, but not on the efficiency. While the review of the dedicated review model is more effective and efficient compared to the review of the behavioral requirements and the functional design, the use of the review model compared to the use of the functional design is more effective, though not more efficient. The modeling language of the review model has thus an impact on the effectiveness of the review but not on the efficiency (at least in case of the proposed review model in ITU MSC notation and the functional design). We can assume that the increase in efficiency is due to merging two specifications in one review model. This finding is in line with findings from another experiment reported in [53], which shows that an additional merging of the bMSCs within the review model has no effect toward effectiveness but can impact the efficiency of the review. However, the threats to validity regarding the construction of our measurement for efficiency must be taken into account.

While the experimental results show that both experienced and unexperienced reviewers are supported by the proposed review model, we actually found no indications that different review models for experienced and unexperienced reviewers might further aid the review. However, all participants in our experiment had some kind of a software engineering background. Hence, our experimental results provide no evidence if the dedicated review model has positive effects on reviews conducted by stakeholders without any kind of software engineering background—to evaluate this is obviously a topic for our future work.

Although our experience indicates that ITU MSCs are well suited for conducting manual reviews of behavioral requirements, our experiments were not designed to validate the suitability of ITU MSC models and notations for reviews. Further research is required to investigate this in more detail. However, based on the experiences gained from the results of the three experiments we assume ITU MSCs a good starting for defining a modeling language for review models. While we found that merging of different specifications (behavioral requirements and functional design) into one review model can aid the efficiency of the review, the work reported in [53] indicates that model merging has some limitations. For instance, the merge of single bMSCs does only increase efficiency in case of minor inconsistencies, but is not recommendable in any situation. To understand these limitations better further empirical investigations are needed.

8 Related work

The approach outlined in Sect. 4 relies on well-established techniques from the state of the art. The related work can be separated into three topics. First, existing works regarding the manual validation of artifacts (in particular of models), i.e., by means of manual reviews, are relevant. Second, there is also related work regarding the automated validation of artifacts. While this paper builds upon the manual validation, also fully automated techniques could be of use to check for consistency between behavioral requirements and the functional design. Third, approaches dealing with model evolution and automated model development could be of use to aid in correcting the original artifacts according to changes in one artifact.

8.1 Manual validation

Manual reviews can be performed to validate the up-to-dateness of the behavioral requirements and the correctness of the functional design. As part of the review, the requirements engineer will typically detect deficiencies in the original models and correct these deficiencies right away. Model-based review techniques have been evaluated as very effective and appropriate (cf. [54, 55]). Specializations such as checklist-based reviews (cf. [56]), defect class-based reviews (cf. [57]), usage-based reviews (cf. [58]) and in particular perspective-based reviews (cf. [24, 59]) were partially evaluated as even more effective. Manual reviews from a requirements engineering perspective can be used to detect deficiencies in behavioral requirements and the functional design in terms of stakeholder intentions. The major disadvantages of such manual techniques are their time-consuming nature and the likely occurrence of errors within the manual tasks.

In particular, perspective-based review (also referred to as perspective-based reading) is a widely accepted inspection technique for requirements specifications (cf. [25]). Some work has been done to apply perspective-based reviews to model-based specifications. For example, in Denger and Ciolkowski [60] describe a defect taxonomy to apply a perspective-based inspection technique to statecharts used in the embedded systems domain. In addition, *Binder* defines a checklist to validate statecharts from a testing perspective in [61]. A more general approach to validate model-based specifications is presented by Travassos et al. in [62]. The approach addresses the consistency between UML diagrams of different types. For this purpose, perspective-based

reviews of scenarios from different traceability perspectives are suggested. In Liu et al. [63] present a formal inspection technique to verify code against functional scenarios that are derived from the original specification. This work is in so far related to our work as we use message sequence charts as a notation for the review model, which is also an appropriate representation of scenarios. Liu et al. conclude that their approach using functional scenarios to validate the correctness of a program is more effective in detecting conceptual defects than normal perspective-based reviews.

8.2 Automated validation

A prerequisite for most fully automated validation techniques is the use of traceability links between the artifacts under consideration. Establishing traceability links between requirements and design artifacts (more precisely: between single elements of requirements artifacts and the resulting elements in the design artifacts) is often seen as a basis for continuous requirements engineering (cf. [64, 65]). Based on traceability links, changes to the functional design can be retraced to the requirements artifacts, and necessary changes to the requirements can be detected easily. This may, for example, be used as a trigger for model evolution techniques. In function-centered engineering processes in industry, the requirements are typically not continuously updated, and the requirements engineer is not involved in changes to the functional design. Hence, traceability-centric approaches on their own cannot determine whether a detected change results from changed stakeholder intentions, or were induced erroneously.

Automated verification techniques aim at checking the correctness of a development artifact or the software. Therefore, a correct reference model is needed. Classical model checking approaches rely on the verification of a Kripke structure against some kind of temporal logic (e.g., [66–68]). In addition, approaches using model checking between two models (of the same model type, e.g., [69], and sometimes also between models of different model types, e.g., [70, 71]) exist as well. While verification techniques can be used to detect deficiencies in one model by means of the other model, fully automated techniques cannot decide whether this deficiency is a deficiency of stakeholder intentions, because automated techniques cannot take undocumented knowledge into account. Furthermore, model checking is only of limited use: single counter examples in temporal logic or as finite state machines neither support the engineers in detecting the inconsistency in the original models nor in correcting the original models (cf. [72]).

As an enhancement of model checking, consistency checking (e.g., [71, 73]) can be used to detect inconsistencies between two models. Simulation is also often used to check for consistency between different executable models. Thereby, it is verified that all execution paths specified in one artifact are also executable in another artifact, and, to ensure full consistency, also vice versa. However, this does not aid in deciding which of the two artifacts represents the undocumented stakeholder intentions best.

8.3 Automated model evolution

Another approach to keep track with changing requirements is model evolution. Since manual approaches are time-consuming and error-prone, automated approaches can be used to update the behavioral requirements due to a changed functional design. Since one-way model transformation approaches (e.g., [19, 74]) result in the loss of the original behavioral requirements, model synchronization techniques can be used to implement bidirectional model evolution (cf. [75]). This is most commonly done by using triple graph grammars (cf. [76, 77]). Current approaches addressing concrete model types deal with synchronizations between different notations of the same model (e.g., [78]) or between models of the same type or of comparable model types (e.g., [29, 79–81]). More general approaches are based on graph structures (e.g., [82, 83]). While model evolution would benefit the development of consistent behavioral requirements and functional design, existing approaches do not consider whether or not a change of the functional design results from changed stakeholder intentions. For example, changing the functional design due to technical issues could contradict the behavioral requirements. Model synchronization would change the behavioral requirements accordingly, which might not be in correspondence with the current stakeholder intentions.

9 Conclusion

In function-centered engineering of embedded systems in industry, typically, different companies (e.g., original equipment manufacturer, first-tier suppliers and second-tier supplier) are involved in the development processes. In such a context, it appears frequently that engineering artifacts (like behavioral requirements and functional design) evolve independently from each other—even though all artifacts should be consistent and appropriately reflect the current stakeholder intentions at any time. As automated approaches can only detect inconsistencies, but not determine whether a certain artifact is correct or not, inconsistencies need to be investigated manually.

In this article, we proposed the use of a dedicated review model to support reviewers in the manual validation of behavioral requirements and functional design against stakeholder intensions. The review model integrates information specified in the behavioral requirements and the functional

design into a single consolidated model. We also reported on three controlled experiments conducted to investigate the suitability of the dedicated review model. The experiments have shown that the review of behavioral requirements and the functional design against stakeholder intensions is significantly more effective, efficient, confidence raising and more supportive when using the dedicated review model compared to using the original specifications of the behavioral requirements and the functional design. The results show that the review model approach significantly supports manual validation—independently from whether the review model is used by rather unexperienced or rather experienced reviewers. Moreover, the results show that the improvement in effectiveness gained during review is even higher for experienced reviewers compared to unexperienced reviewers.

When analyzing the results of the three experiments together, one can conclude that both the consolidation of the behavioral requirements and functional design by a dedicated model and to the use of ITU MSCs as modeling language for the review model contribute to the benefits gained. Our future work will focus on comparing the effects of the different modeling language and notations for representing integrated artifacts within a review model. In addition, we will investigate the creation of similar models to support the manual validation of further development artifacts (e.g., real-time requirements, technical architectures and deployment models) for embedded and cyber-physical systems.

Acknowledgements Funding was partially provided by the German Federal Ministry of Education and Research (BMBF) under grant number 01IS12005C (SPES_XT) and 01IS16043V (CrESt).

References

1. Schätz, B., Pretschner, A., Huber, F., Philipps, J.: Model-based development of embedded systems. In: Advances in Object-Oriented Information Systems, OOIS 2002 Workshops, Montpellier, France, 2 Sept 2002, Proceedings, vol. 2426, pp. 298–312 (2002)
2. France, R.B., Rumpe, B.: Model-driven development of complex software: a research roadmap. In: International Conference on Software Engineering, ISCE 2007, Workshop on the Future of Software Engineering, FOSE 2007, 23–25 May 2007, Minneapolis, MN, USA, pp. 37–54 (2007)
3. Schmidt, D.C.: Guest editor's introduction: model-driven engineering. IEEE Comput. **39**(2), 25–31 (2006)
4. ISO 26262-1: Road vehicles—functional safety—part 1: vocabulary (2011)
5. SAE International Standard 4761: Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment. SAE (1996)
6. ISO/IEC/IEEE 24765: ISO/IEC/IEEE international standard—systems and software engineering: vocabulary, Aug 2017
7. Pretschner, A., Broy, M., Krüger, I.H., Stauner, T.: Software engineering for automotive systems: a roadmap. In: International Conference on Software Engineering, ISCE 2007, Workshop on the Future of Software Engineering, FOSE 2007, 23–25 May 2007, Minneapolis, MN, USA, pp. 55–71 (2007)
8. Jantsch, A., Sander, I.: On the roles of functions and objects in system specification. In: Proceedings of the Eighth International Workshop on Hardware/Software Codesign, CODES 2000, San Diego, California, USA, 2000, pp. 8–12 (2000)
9. Daun, M., Höfflinger, J., Weyer, T.: 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 Apr 2014, pp. 226–234 (2014)
10. ISO/IEC/IEEE systems and software engineering—architecture description. ISO/IEC/IEEE 42010:2011(E) (revision of ISO/IEC 42010:2007 and IEEE Std 1471–2000), pp. 1–46, Dec 2011
11. ISO/IEC TS 24748-1: Systems and software engineering—life cycle management—part 1: guidelines for life cycle management (2016)
12. Nuseibeh, B.: Weaving together requirements and architectures. IEEE Comput. **34**(3), 115–117 (2001)
13. Whalen, M.W., Murugesan, A., Heimdahl, M.P.E.: Your what is my how: why requirements and architectural design should be iterative. In: First IEEE International Workshop on the Twin Peaks of Requirements and Architecture, TwinPeaks@RE 2012, Chicago, IL, USA, 25 Sept 2012, pp. 36–40 (2012)
14. ISO/IEC/IEEE International Standard—systems and software engineering—life cycle processes: requirements engineering. ISO/IEC/IEEE 29148:2011(E), pp. 1–94, Dec (2011)
15. DOT/FAA/AR-08/32: Requirements Engineering Management Handbook. U.S. Department of Transportation, Federal Aviation Administration, Springfield, Virginia, United States (2009)
16. Jeon, S.-U., Hong, J.-E., Bae, D.-H.: Interaction-based behavior modeling of embedded software using UML 2.0. In: Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'06), p. 5 (2006)
17. Weber, M., Weisbrod, J.: Requirements engineering in automotive development experiences and challenges. In: 10th Anniversary IEEE Joint International Conference on Requirements Engineering (RE 2002), 9–13 Sept 2002, Essen, Germany, pp. 331–340 (2002)
18. ITU-T Z.120: Recommendation ITU-T Z.120: Message Sequence Chart (MSC). International Telecommunication Union (2011)
19. Whittle, J., Jayaraman, P.K.: Synthesizing hierarchical state machines from expressive scenario descriptions. ACM Trans. Softw. Eng. Methodol. **19**(3), 1–45 (2010)
20. de Alfaro, L., Henzinger, T.A.: Interface automata. In: Proceedings of the 8th European Software Engineering Conference Held Jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering, New York, NY, USA, pp. 109–120 (2001)
21. Albers, K., et al.: System function networks. In: Pohl, K., Broy, M., Daembkes, H., Hönninger, H. (eds.) Advanced Model-Based Engineering of Embedded Systems, Extensions of the SPES 2020 Methodology, pp. 119–144. Springer, Berlin (2016)
22. Leveson, N.G.: Safeware—System Safety and Computers: A Guide to Preventing Accidents and Losses Caused by Technology. Addison-Wesley, Reading (1995)
23. SAE International Standard J1239_200901: Potential failure mode and effects analysis in design (design FMEA), potential failure mode and effects analysis in manufacturing and assembly processes (process FMEA) (2009)
24. Basili, V.R., et al.: The empirical investigation of perspective-based reading. Empir. Softw. Eng. **1**(2), 133–164 (1996)

25. Shull, F., Rus, I., Basili, V.R.: How perspective-based reading can improve requirements inspections. IEEE Comput. **33**(7), 73–79 (2000)
26. Alur, R., Etessami, K., Yannakakis, M.: Inference of message sequence charts. IEEE Trans. Softw. Eng. **29**(7), 623–633 (2003)
27. Uchitel, S., Kramer, J., Magee, J.: Synthesis of behavioral models from scenarios. IEEE Trans. Softw. Eng. **29**(2), 99–115 (2003)
28. Uchitel, S., Brunet, G., Chechik, M.: Behaviour model synthesis from properties and scenarios. Presented at the 29th International Conference on Software Engineering, 2007. ICSE 2007, pp. 34–43 (2007)
29. Uchitel, S., Brunet, G., Chechik, M.: Synthesis of partial behavior models from properties and scenarios. IEEE Trans. Softw. Eng. **35**(3), 384–406 (2009)
30. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation, 3rd edn. Boston, Pearson (2007)
31. Harel, D., Segall, I.: Synthesis from scenario-based specifications. J. Comput. Syst. Sci. **78**(3), 970–980 (2012)
32. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation, 1st edn. Addison-Wesley Publishing Company, Reading (1979)
33. Nerode, A.: Linear automaton transformations. Proc. Am. Math. Soc. **9**(4), 541–544 (1958)
34. Brauer, W.: Automatentheorie: Eine Einführung in die Theorie endlicher Automaten. Softcover reprint of the original 1st edn. 1984. Vieweg + Teubner Verlag, Stuttgart (1984)
35. Sikora, E., Daun, M., Pohl, K.: Supporting the consistent specification of scenarios across multiple abstraction levels. In: Requirements Engineering: Foundation for Software Quality, 16th International Working Conference, REFSQ 2010, Essen, Germany, 30 June, 2 July 2010. Proceedings, vol. 6182, pp. 45–59 (2010)
36. Nolte, S.: QVT Operational Mappings: Modellierung mit der Query Views Transformation. Springer, Berlin (2010)
37. OMG: Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. Object Management Group (2015)
38. Daun, M., Salmon, A., Weyer, T., Pohl, K.: The impact of students' skills and experiences on empirical results: a controlled experiment with undergraduate and graduate students. In: Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering, EASE 2015, Nanjing, China, 27–29 Apr 2015, p. Paper 29 (2015)
39. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering, 2012th edn. Springer, New York (2012)
40. Salman, I., Misirli, A.T., Juzgado, N.J.: Are students representatives of professionals in software engineering experiments?. In: 37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, 16–24 May 2015, vol. 1, pp. 666–676 (2015)
41. Siegmund, J., Siegmund, N., Apel, S.: Views on internal and external validity in empirical software engineering. In: 37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, 16–24 May 2015, vol. 1, pp. 9–19 (2015)
42. Berander, P.: Using students as subjects in requirements prioritization. In: 2004 International Symposium on Empirical Software Engineering (ISESE 2004), 19–20 Aug 2004, Redondo Beach, CA, USA, pp. 167–176 (2004)
43. Höst, M., Regnell, B., Wohlin, C.: Using students as subjects: a comparative study of students and professionals in lead-time impact assessment. Empir. Softw. Eng. **5**(3), 201–214 (2000)
44. Svahnberg, M., Aurum, A., Wohlin, C.: Using students as subjects an empirical evaluation. In: Proceedings of the Second International Symposium on Empirical Software Engineering and Measurement, ESEM 2008, 9–10 Oct 2008, Kaiserslautern, Germany, pp. 288–290 (2008)
45. Tichy, W.F.: Hints for reviewing empirical work in software engineering. Empir. Softw. Eng. **5**(4), 309–312 (2000)
46. Venkatesh, V., Bala, H.: Technology acceptance model 3 and a research agenda on interventions. Decis. Sci. **39**(2), 273–315 (2008)
47. Jedlitschka, A., Ciolkowski, M., Pfahl, D.: Reporting experiments in software engineering. In: Shull, F., Singer, J., Sjøberg, D.I.K. (eds.) Guide to Advanced Empirical Software Engineering, pp. 201–228. Springer, London (2008)
48. Ramsey, P.H.: Exact type 1 error rates for robustness of student's t test with unequal variances. J. Educ. Behav. Stat. **5**(4), 337–349 (1980)
49. Arcuri, A., Briand, L.C.: A practical guide for using statistical tests to assess randomized algorithms in software engineering. In: Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011, Waikiki, Honolulu, HI, USA, 21–28 May 2011, pp. 1–10 (2011)
50. Kitchenham, B.: Robust statistical methods: why, what and how: keynote. In: Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering, EASE 2015, Nanjing, China, 27–29 Apr 2015, p. Paper 1 (2015)
51. Faul, F., Erdfelder, E., Lang, A., Buchner, A.: G*Power 3: a flexible statistical power analysis program for social, behavioral, and biomedical sciences. Behav. Res. Methods **39**(2), 175–191 (2007)
52. Campbell, D.T., Stanley, J.C.: Experimental and Quasi-Experimental Designs for Research. Houghton Mifflin, Boston (1963)
53. Daun, M., Brings, J., Weyer, T.: On the impact of the model-based representation of inconsistencies to manual reviews. In: Conceptual Modeling, pp. 466–473 (2017)
54. Boehm, B.W., Basili, V.R.: Software defect reduction top 10 list. IEEE Comput. **34**(1), 135–137 (2001)
55. Gilb, T., Graham, D., Finzi, S.: Software Inspection. Addison-Wesley, Wokingham (1993)
56. Fagan, M.E.: Advances in software inspections. IEEE Trans. Softw. Eng. **12**(7), 744–751 (1986)
57. Porter, A.A., Votta, L.G., Basili, V.R.: Comparing detection methods for software requirements inspections: a replicated experiment. IEEE Trans. Softw. Eng. **21**(6), 563–575 (1995)
58. Abdelnabi, Z., Cantone, G., Ciolkowski, M., Rombach, H.D.: Comparing code reading techniques applied to object-oriented software frameworks with regard to effectiveness and defect detection rate. In: 2004 International Symposium on Empirical Software Engineering (ISESE 2004), 19–20 Aug 2004, Redondo Beach, CA, USA, pp. 239–248 (2004)
59. Shull, F., et al.: What we have learned about fighting defects. In: 8th IEEE International Software Metrics Symposium (METRICS 2002), 4–7 June 2002, Ottawa, Canada, pp. 249–258 (2002)
60. Denger, C., Ciolkowski, M.: High quality statecharts through tailored, perspective-based inspections. In: 2003 Proceedings 29th Euromicro Conference, pp. 316–323 (2003)
61. Binder, R.V.: Testing Object-Oriented Systems: Models, Patterns, and Tools. Addison-Wesley, Reading (1999)
62. Travassos, G., Shull, F., Fredericks, M., Basili, V.R.: Detecting defects in object-oriented designs: using reading techniques to increase software quality. In: Proceedings of the 1999 ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages & Applications (OOPSLA'99), Denver, Colorado, USA, 1–5 Nov 1999, pp. 47–56 (1999)
63. Liu, S., Chen, Y., Nagoya, F., McDermid, J.A.: Formal specification-based inspection for verification of programs. IEEE Trans. Softw. Eng. **38**(5), 1100–1122 (2012)
64. Gotel, O.C.Z., Finkelstein, A.: An analysis of the requirements traceability problem. In: Proceedings of the First IEEE International Conference on Requirements Engineering, ICRE'94, Colorado Springs, Colorado, USA, 18–21 Apr 1994, pp. 94–101 (1994)

65. Winkler, S., von Pilgrim, J.: A survey of traceability in requirements engineering and model-driven development. Softw. Syst. Model. **9**(4), 529–565 (2010)
66. Clarke, E.M., Emerson, E.A., Sifakis, J.: Model checking: algorithmic verification and debugging. Commun. ACM **52**(11), 74–84 (2009)
67. Larsen, K.G.: Efficient local correctness checking. In: Computer Aided Verification, Fourth International Workshop, CAV'92, Montreal, Canada, 29 June 1 July 1992, Proceedings, vol. 663, pp. 30–43 (1993)
68. Holzmann, G.J.: The model checker SPIN. IEEE Trans. Softw. Eng. **23**(5), 279–295 (1997)
69. Blanc, X., Mounier, I., Mougenot, A., Mens, T.: Detecting model inconsistency through operation-based model construction. In: 30th International Conference on Software Engineering (ICSE 2008), Leipzig, Germany, 10–18 May 2008, pp. 511–520 (2008)
70. Grundy, J.C., Hosking, J.G., Mugridge, W.B.: Inconsistency management for multiple-view software development environments. IEEE Trans. Softw. Eng. **24**(11), 960–981 (1998)
71. Fradet, P., Le Métayer, D., Périn, M.: Consistency checking for multiple view software architectures. In: Software Engineering ESEC/FSE'99, 7th European Software Engineering Conference, Held Jointly with the 7th ACM SIGSOFT Symposium on the Foundations of Software Engineering, Toulouse, France, Sept 1999, Proceedings, vol. 1687, pp. 410–428 (1999)
72. Borges, R.V., d'Avila Garcez, A.S., Lamb, L.C.: Integrating model verification and self-adaptation. In: ASE 2010, 25th IEEE/ACM International Conference on Automated Software Engineering, Antwerp, Belgium, 20–24 Sept 2010, pp. 317–320 (2010)
73. Paige, R.F., Brooke, P.J., Ostroff, J.S.: Metamodel-based model conformance and multiview consistency checking. ACM Trans. Softw. Eng. Methodol. **16**(3), 11 (2007)
74. Milicev, D.: Automatic model transformations using extended UML object diagrams in modeling environments. IEEE Trans. Softw. Eng. **28**(4), 413–430 (2002)
75. Giese, H., Wagner, R.: From model transformation to incremental bidirectional model synchronization. Softw. Syst. Model. **8**(1), 21–43 (2009)
76. Giese, H., Wagner, R.: Incremental model synchronization with triple graph grammars. In: Model Driven Engineering Languages and Systems, 9th International Conference, MoDELS 2006, Genova, Italy, 1–6 Oct 2006, Proceedings, vol. 4199, pp. 543–557 (2006)
77. Hermann, F., et al.: Model synchronization based on triple graph grammars: correctness, completeness and invertibility. Softw. Syst. Model. **14**(1), 241–269 (2015)
78. Giese, H., Hildebrandt, S., Neumann, S.: Model synchronization at work: keeping SysML and AUTOSAR models consistent. In: Graph Transformations and Model-Driven Engineering Essays Dedicated to Manfred Nagl on the Occasion of his 65th Birthday, vol. 5765, pp. 555–579 (2010)
79. Damas, C., Lambeau, B., Roucoux, F., van Lamsweerde, A.: Analyzing critical process models through behavior model synthesis. In: 31st International Conference on Software Engineering, ICSE 2009, 16–24 May 2009, Vancouver, Canada, Proceedings, pp. 441–451 (2009)
80. van Paesschen, E., Meuter, W.D., D'Hondt, M.: SelfSync: a dynamic round-trip engineering environment. In: Model Driven Engineering Languages and Systems, 8th International Conference, MoDELS 2005, Montego Bay, Jamaica, 2–7 Oct 2005, Proceedings, vol. 3713, pp. 633–647 (2005)
81. Malavolta, I., Muccini, H., Pelliccione, P., Tamburri, D.A.: Providing architectural languages and tools interoperability through model transformation technologies. IEEE Trans. Softw. Eng. **36**(1), 119–140 (2010)
82. Kimelman, D., Kimelman, M., Mandelin, D., Yellin, D.M.: Bayesian approaches to matching architectural diagrams. IEEE Trans. Softw. Eng. **36**(2), 248–274 (2010)
83. Agrawal, A.: Graph rewriting and transformation (GReAT): a solution for the model integrated computing (MIC) bottleneck. In: 18th IEEE International Conference on Automated Software Engineering (ASE 2003), 6–10 Oct 2003, Montreal, Canada, pp. 364–368 (2003)

Marian Daun is a researcher at paluno - the Ruhr Institute for Software Technology, University of Duisburg-Essen. His main research interests are in the area of model-based software engineering, empirical software engineering and software engineering education and training. In various national and international publicly funded projects as well as in industry co-operations, he works on model-based engineering techniques to foster the structured development of embedded and cyber-physical systems. He earned a Ph.D. in computer science and holds a diploma degree in business information systems. He has been an author or co-author of more than 40 peer-reviewed journal, conference and workshop publications. He has acted and acts as organizer, program committee member and reviewer for international scientific conferences, workshops and journals.

Thorsten Weyer is head of the group "Requirements Engineering and Conceptual Design" at The Ruhr Institute for Software Technology (paluno). For many years now, he has been contributing his expertise in the development and evaluation of engineering methods for software-intensive systems in close cooperation with both leading international companies and small- and medium-sized enterprises from the high-technology sector. His current personal research focus is on engineering methods for autonomous and collaborative cyber-physical systems (CPS), which form the technological basis of many future scenarios such as autonomous driving and Industrie 4.0. Thorsten Weyer is the author of numerous international scientific publications and has made major contributions to popular textbooks. Furthermore, as a full member of the International Requirements Engineering Board (IREB) and member of the IREB Council, he has an important position in the international standardization of the requirements engineering practice.

Klaus Pohl is a full professor for software systems engineering at the University of Duisburg-Essen and director of paluno – The Ruhr Institute for Software Technology. From 2005 to 2007, he was the scientific founding director of the Irish Software Engineering Research Centre (lero). His research interests include adaptive systems, digitization/big data, cloud computing, cyber-physical systems, service orientation, variability management and requirements engineering. He is/was coordinator of many research projects and member of scientific and industrial steering committees, including the European Technology Platform NESSI, the European BDVA and the German Innovation Alliance SPES. Klaus Pohl is/was program and general chair of several international and national conferences, including the 29th Intl. Conference on Advanced Information Systems Engineering (CAiSE 2017), 35th Intl. Conference on Software Engineering (ICSE 2013), 9th and 12th Intl. Software Product Line Conference (SPLC 2005/2008), 18th Intl. Conference on Advanced Information Systems Engineering (CAiSE 2006), German GI Software Engineering Conference 2005, Intl. Requirements Engineering Conference (RE 2002). He is (co-)author of more than 300 peer-reviewed publications and six text books.