Function-centered Engineering of Embedded Systems
Evaluating Industry Needs and Possible Solutions

Marian Daun[1], Jens Höfflinger[2] and Thorsten Weyer[1]
[1]*paluno- The Ruhr Institute for Software Technology, University of Duisburg-Essen, 45127 Essen, Germany*
[2]*Robert Bosch GmbH, 70442 Stuttgart, Germany*

Keywords: Automotive Engineering, Early Design, Embedded Systems, Functional Design, Function-centered Engineering.

Abstract: Research in engineering disciplines has to keep track of current developments and challenges in industry to provide adequate solutions. As function-centered engineering of embedded systems is commonly used in industry to cope with several challenges (e.g., to deal with the increasing number of functions realized in software rather than in hardware, to reduce redundancy among functions implemented, to reduce the number of cost-intensive electronic control units and sensors, or to foster re-use of developed functions in multiple systems and environments), it is of importance for research to identify challenges and needs arising from function-centered engineering and to provide fitting solution concepts. To identify these industry needs, we recently conducted a study among German embedded industry. We used a combination of different investigation techniques (i.e. questionnaires, workshops and expert interviews, and case studies) to identify, concretize, and verify industry needs. Furthermore, possible solution ideas were developed and evaluated to (i) check appropriateness of identified needs and to gain more insights, as well as to (ii) provide first solution concepts suitable for industry. This paper discusses the evaluation method, the major results and the current state of the solution approach.

1 INTRODUCTION

Embedded systems are highly integrated systems consisting of hardware and software parts. The software part of embedded systems is increasing and more and more functionality is realized by software (Broy. 2006). For example, the car's engine temperature is no longer measured using a specific sensor. Nowadays, it is predicted using existing distant sensors and algorithms. Within this evolution, functionality is not only bound to single software functions deployed on single control units. Functionality is realized through the interplay of different functions, even of different control units. For example, passenger protection in crash situations is realized by the interplay of: the door control unit (to unlock the doors), the body control module (to use the turn signals to indicate an accident), the engine control unit (to stop the engine), and, of course, the airbags.

To design this functional interplay explicitly, function-centered engineering (see e.g. (Pretschner et al. 2007)) is commonly used. In function-centered engineering, the system functions play a major role. Therefore, a first functional design is created early in the development process, right after the behavioral requirements for the system have been specified (see Figure 1). A functional design usually consists of three types of artifacts: a function network which defines the interactions and dependencies between different functions, a function hierarchy which structures the system functions in terms of their sub-functions, and diagrams defining the behavior of each system function. The functional design then serves as a basis for subsequent development activities like defining the electric and electronic architecture or designing the deployment architecture.

To identify current industry needs and to meet these needs by suitable solutions, we conducted a study among German industry with regard to function-centered engineering of embedded systems. The study was conducted in 2012/2013, participants stem from large worldwide operating original equipment manufacturers and suppliers. The study was qualitative in nature with minor quantitative parts to gain first insights. Therefore, we used questionnaires, workshops and expert interviews, and case studies. In addition, solution approaches were developed to

Daun M., Höfflinger J. and Weyer T..
Function-centered Engineering of Embedded Systems - Evaluating Industry Needs and Possible Solutions.
DOI: 10.5220/0004967202260234
In *Proceedings of the 9th International Conference on Evaluation of Novel Approaches to Software Engineering* (ENASE-2014), pages 226-234
ISBN: 978-989-758-030-7

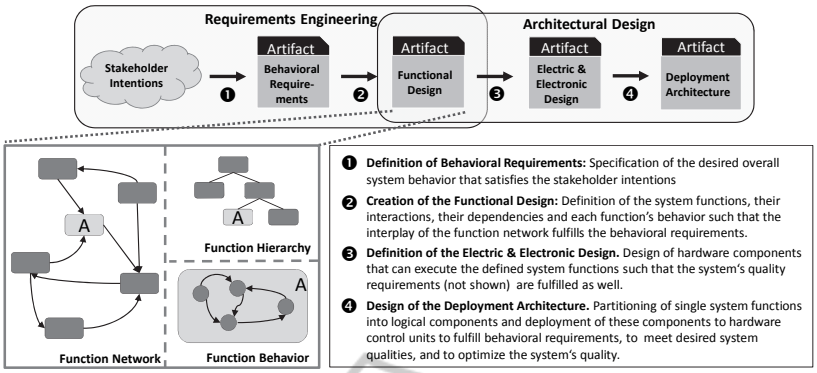


Figure 1: Function-Centered Engineering Process and its Main Artifacts.

verify revealed needs, to gain further insights, and to ensure suitability and appropriateness for industrial usage. The study design is described in detail in Section 2.

Section 3 discusses our main findings and the current state of the solution approach: Automated techniques for correctness checking and co-evolution of engineering artifacts are desired to support function-centered engineering. Albeit automated techniques are strongly desired, fully automated approaches are often not applicable to industry (e.g. because in the development of safety-critical systems decisions cannot be made in fully automated manner). As solution the use of dedicated review models for correctness checking and model evolution seems appropriate. These models can be created and processed in fully automated manner to support manual decision-making by the responsible engineer. In addition, if desired, further automated techniques can be applied to aid the manual review.

In the end Section 4 concludes this paper and discusses the generalizability of the results.

2 STUDY DESIGN

The study was designed to gain qualitative insights into the current state of practice and industry needs. Therefore, several techniques like questionnaires, workshops and case studies were used to address the following research questions:

RQ-1 Determine the current state of practice.

RQ-2 Identify problems and room for improvement within the current state of practice.

RQ-3 Determine how the industry needs can be solved by the use of current techniques, and which enhancements are necessary.

Section 2.1 first introduces the participating companies. Section 2.2 details the investigative method and Section 2.3 presents the techniques used.

2.1 Participants

The study was conducted among companies involved in the German SPES 2020 XTCore project. The participating companies have their main focus of interest within the development of automotive or avionic systems, are internationally operating, and can be considered as large original equipment manufacturers or suppliers. The participants were chosen by the companies and stem from research and productive units. Most participants are working for at least 5 years within the specific unit of the company. They take part in the development process as requirements engineer, function designer, architect, or department/project leader.

2.2 Investigative Method

We performed our investigations in three steps (see Figure 2). First, we developed an initial questionnaire to gain preliminary insights into the current state of practice and into industry needs. Based on the first results from the questionnaire, we conducted interviews with the participants as well as workshops to elaborate on the first results. Second, we developed a solution approach to address the industry needs that were revealed during the first step. Third, we presented the solution approach to the participants and conducted workshops and interviews again to gain more insights into their needs. We used this newly gained knowledge to revise and improve the solution approach. Besides, we also applied our solution approach to industrial case studies and presented them also to the participants. Based on the feedback from the participants, we once again concretized, revised, and enhanced our knowledge about industry needs.

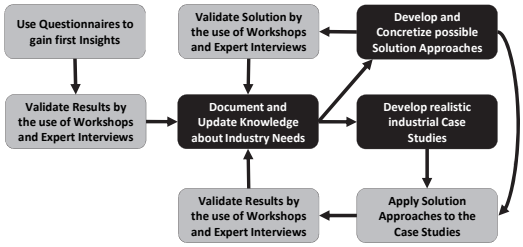


Figure 2: Used investigation techniques and results.

2.3 Used Elicitation Techniques

Questionnaire. The participants to the initial questionnaire were chosen and invited by their companies. The questionnaire itself was anonymous to encourage real and unsugarcoated answers. This prohibits drawing conclusions regarding a single company. In total, 9 professionals participated in the initial questionnaire.

The questionnaire was designed to get first insights into current state of practice (RQ-1) and into industry needs (RQ-2). Therefore, mostly half-open questions were used to ensure all of participants' answers could be sketched. Furthermore, 4-point or 6-point Likert scale items were used to elicit opinions on industry needs and current state of practice. This was chosen to force participants to make a decision in terms of yes or no, good or bad, and not allowing neutral answers. Also open questions were used to give the participants the possibility to comment on their answers and to discuss the understanding of common used vocabulary. The questionnaire itself was designed in iterative evolution by us, other academic partners, and industry partners within the SPES project. For final quality assurance, empirical experts from the Fraunhofer Institute for Empirical Software Engineering conducted a final review of the questionnaire and provided suggestions.

Workshops and Expert Interviews. Experts participating in interviews and workshops were also chosen by the companies. Since questionnaires were conducted anonymously, there is no knowledge about the relationship between participants in the questionnaire and participants in the interviews and workshops. Most participants in the interviews or discussion rounds were advanced engineers, with knowledge gained by years of work within the company in different roles and projects. To reduce iteration cycles between local workshops, regular telephone conferences were held. This setting allowed for fast feedback as well as fast rework of case studies and of the solution approach.

Workshops and expert interviews were conducted to validate and to detail the results taken from the questionnaires. Furthermore, workshops were conducted to evaluate possible solutions and the results from the case studies. Thereby, the workshops addressed RQ-1, RQ-2, and also RQ-3.

Solution Approach. A solution approach was developed to support the identification of industry needs (RQ-2) and to gain insights whether industry needs may be solved by the current state of the art (RQ-3). It revealed that the current state of the art could not solve all industry needs. Therefore, we developed our own initial solution approach based on the state of the art. This initial solution approach has been evaluated in workshops as appropriate and will be enhanced in future work.

Case Studies. All case studies are typical representatives of embedded systems of the automotive and the avionic domain. The gained industry needs were used to develop adequate case studies, for example, of a lane keeping support, a parking assist system, or a collision avoidance system. All case studies were used to validate and concretize revealed industry needs (RQ-2). Beside RQ-2, case studies also addressed RQ-3 by examining whether the developed solution approach is appropriate.

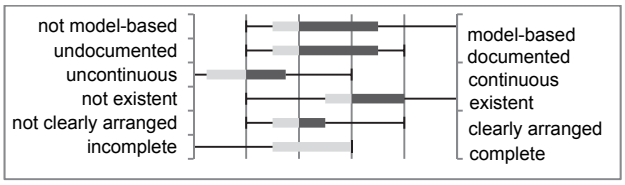
Development of the case studies according to industry needs and the application of the solution approach to the case studies were conducted in cooperation between us, other academic partners, and other industry partners.

3 RESULTS

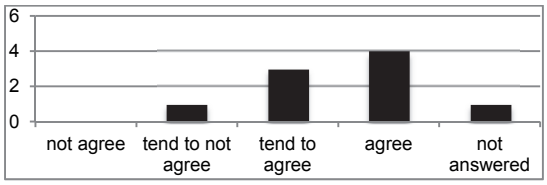
During the study two major needs were identified: (i) the need for continuous engineering (Section 3.1) and (ii) the need to ensure correctness of the functional design (Section 3.2). In addition, needs regarding the model usage in function-centered engineering were unveiled (Section 3.3). These documentation formats result from the desire for continuous engineering and correctness of the functional design. They were also needed to develop a suitable solution approach to meet industry needs. Section 3.4 presents the current state of the proposed solution approach.

3.1 Continuous Development

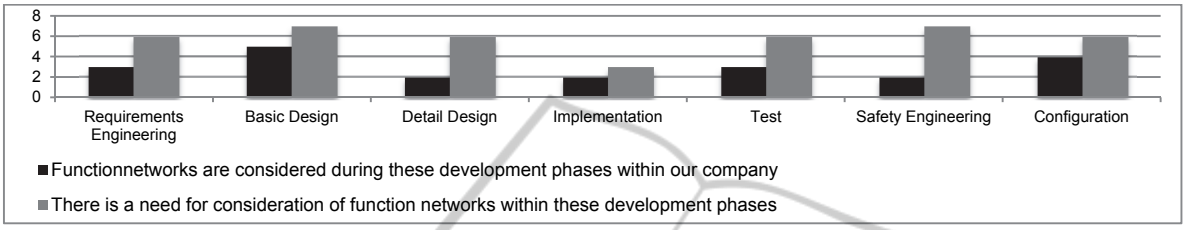
We investigated industry needs regarding the development process of function-centered engineering. In a first step, we evaluated the current situation of development processes used within the participants' companies. As Figure 3 (a) depicts, in general the



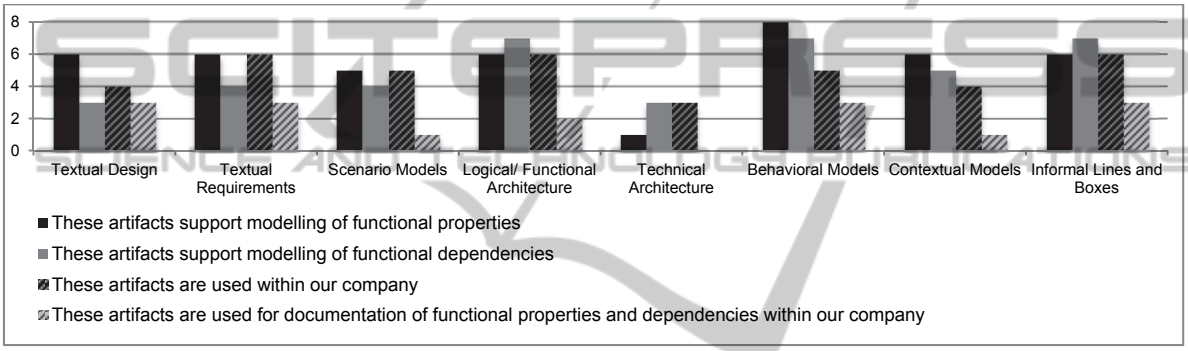
(a) The current development process within our company is…



(b) There is a need to develop an approach addressing the continuous functional analysis of functional properties and functional dependencies



(c) Consideration of function networks during development phases – current situation and needed situation



(d) Artifacts used and needed to document functional properties and functional dependencies

Figure 3: Results from the Questionnaires.

development process itself seems to be existent (i.e. a systematic development process is applied and pursued). It can also be recognized that model-based development is used within the industry, as well as explicit documentation seems to be rather common. It becomes also clear, that most development processes are rather uncontinuous. In addition, the results in Figure 3 (b) suggest that there is an intense industry need to support continuous development within function-centered engineering. This was approved by the interviews and workshops.

To concretize the need regarding specific phases, we asked the participants whether function networks are considered during single development phases within their companies and whether they think that function networks should be considered in the single phases. As the results depicted in Figure 3 (c) indicate, function networks are currently mainly considered in basic design (i.e. the functional design is built during basic design), and in configuration (i.e. to derive single products from software product lines). As shown, the most room for improvement is seen regarding requirements engineering, detail design (phase which develops the technical architecture), safety engineering and testing.

In subsequent workshops and expert discussions industry partners agreed on three major needs to support the continuous development in function-centered engineering:

- Automated support for the evolution of the functional design from behavioral requirements, and for updating the functional design, due to changes in the requirements specification.
- Automated support for change propagation from functional design back to the requirements. This is due to the fact, that different responsible engineers evolve the functional design and behavioral requirements after their creation. These tasks are often independent from each other and lead to inconsistencies, which affect the correctness.
- Automated support to develop artifacts in the later phases. Since the functional design serves as the major artifact of function-centered engineering, all subsequent artifacts are based on the functional design.

Solution Idea. To support the continuous development in early phases, common model transformation techniques (e.g. (Milicev. 2002)) may be used. These approaches are adapted to fit the transformation from behavioral requirements to functional design. Albeit, this seems to be an appropriate support to develop an initial consistent functional design, it does not support the continuous development in later phases (when the behavioral requirements and the functional design are already existent). Participants claimed that a fully automated approach (e.g. by use of model synchronization techniques like (Giese and Wagner. 2006) or (Hermann et al., 2013)) is desired, but not applicable. This is due to parallel development: it is common in industry that different specifications are reworked at the same time. This means that changes may be applied to the behavioral requirements, while at the same time other changes are applied to the functional design. Automated techniques cannot resolve inconsistencies in this case, since it is not decidable, which elements are correct. To address this issue the use of dedicated review models seems appropriate. These models are generated by automated model transformations and reviewed and revised by the responsible engineer. Note that, these revisions may lead to necessary changes to the source model as well. These changes once again are propagated by the use of another review model.

To support the development of artifacts of the later phases, such as the electric and electronic architecture or the deployment architecture, existing automated techniques can be adopted. Therefore, graph analysis techniques (e.g. (Cox et al., 2001)), or pattern matching techniques (e.g. (Beyer et al., 2005), or (Gross and Yu, 2001)) seem appropriate.

3.2 Correctness of the Functional Design

Discussions regarding the need for continuous development and its possible solution pointed out that the functional design is the central engineering artifact and is connected to nearly all other artifacts such as the behavioral requirements, the electric and electronic design and the deployment architecture. Therefore, industry professionals stressed the need to ensure the correctness of the functional design.

In particular automated support for detection and correction of deficiencies is needed. Following circumstances were described as major sources of an incorrect functional design:

- Changes to the behavioral requirements.
- Changed stakeholder intentions, which have not been documented within the behavioral requirements. These typically arise during the evolution of the functional design and concretizations discussed with the stakeholders.
- Unspecified behavior resulting from functional interplay, which arises from the combination of multiple functions. The single functions behaviors create new, unspecified behavior in their interplay, much akin to feature interactions.

Solution Idea. Like supporting the continuous development, the problem of decidability whether the behavioral requirements or the functional design is correct, prohibits the use of automated correctness-checking techniques like (Clarke et al., 2009) or (Holzmann, 1997). In addition, stakeholder intentions may change due to realization decisions and thereby both artifacts (the functional design and the behavioral requirements) may become outdated. Another reason, why current correctness-checking techniques are not applicable is that they provide only one single counterexample that is not related to the original models. According to the participants, this is also inadequate for industrial use. This finding is consistent with shortcomings of model checking described by (Borges et al., 2010).

In contrast manual review approaches seem to be very effective in many studies (e.g. (Boehm and Basili. 2001); (Gilb and Graham. 1993)). Especially perspective-based reviews seem to be promising (cf. e.g. (Shull et al. 2002)). In discussions, the participants also stated that they think perspective-based reviews could be helpful to check the correctness of the functional design. By application of this technique to the case studies, some deficiencies were unveiled: e.g. manual reviews of both artifacts (behavioral requirements and functional design) did not necessarily ensure consistency between them, and architects reviewing the requirements and requirements engineers reviewing the functional design were not familiar with the documentation languages in which the respective artifacts were documented.

To this end it was agreed, that a dedicated review model supports the correctness checking of the functional design best. The review model can be derived by automated model transformations from behavioral requirements and functional design. Thereafter, manual and automated reviews can be performed to ensure the correctness of the review model. By fully automated back transformations of the review model changes can be propagated and the correctness be ensured. As the use of a review model is also suggested to address continuous function-centered engineering, this is convenient to increase acceptability, since model evolution and correctness checking can

be conducted within one step.

3.3 Model Usage

As the functional design is the central development artifact and the correctness of the functional design is related to the documented behavioral requirements, requirements regarding both artifacts were elicited. To determine which model types are necessary to document requirements and the functional design, we questioned the possible benefit of model types to support the documentation of functional properties and functional dependencies. We also asked about the current overall usage of these artifact types within the interviewees' companies and their current usage to support documentation of functional properties and functional dependencies. The results are depicted in Figure 3 (d). By further interviews, these results have been concretized.

Behavioral Requirements. Currently, textual requirements, scenario models, behavior models and context models are mainly used for specifying embedded software requirements in practice. All of them seem to be appropriate to support the documentation of functional properties and functional dependencies. But, neither of them is used to support the documentation of functional properties and dependencies in practice. By further discussions with domain experts we gained the insight that this is once due to the use of partial models in different languages and once due to the fact that many functional dependencies result from design decisions not taken during requirements engineering.

Use of interviews and applications to the case studies gave rise to the comprehension that the most valued requirements for embedded systems are behavioral requirements. To be more precise, especially interaction-based behavioral requirements models (e.g. sequence diagrams) have been proven useful in the context of embedded systems. This is due to their capabilities in discussing stakeholder intentions (which are well known from scenario-based requirements engineering) and due to their ability to focus on the externally visible behavior by means of interactions. The latter is of special interest to agree on the interfaces of the embedded software and to separate the software under development from its context.

As Figure 3 (d) shows, textual requirements are used and seem to be able to support the documentation of functional properties and dependencies. By further discussions it was revealed that textual requirements are mostly used due to contractual needs between supplier and integrator or supplier and subsupplier. From application of the case studies and further investigations, it was agreed that textual requirements as well as textual design documents do not support function-centered engineering adequately: textual requirements and design artifacts lack formalism, which is needed to support industry needs regarding automated support for continuous development and formal proof of correctness.

Solution Idea. As discussed, behavioral requirements are defined to distinguish between the system and its context, to briefly sketch the system's interfaces in terms of message exchanges and to describe the intended system behavior in terms of inputs and outputs. To document such behavioral requirements, interaction-based models are appropriate. In the engineering of embedded software, message sequence charts (ITU. 2011) are commonly used for this purpose (Weber and Weisbrod, 2002). Therefore, the participants of our study agreed on using message sequence charts for documenting behavioral requirements. Their formal semantics (see (ITU, 2011), (Mauw and Reniers, 1999), and (Hélouët and Maigat, 2001)) adequately support the usage of automated techniques.

Functional Design. As Figure 3 (d) depicts, behavioral models, logical/structural models and context aspects seem to be of specific importance to the function-centered engineering process. Since the functional design is the central artifact of function-centered engineering, it is important to keep the functional design self-contained. Workshops, interviews and applications to the case studies revealed following requirements for the desired documentation format of the functional design.

- Specify the behavior of system functions. Furthermore, not only the behavior of single system functions has to be defined, there is a specific need to reckon the functional behavior resulting from the interplay of different system functions.
- Document the structure of functional dependencies. For example, these dependencies are needed to analyze the functional design, to detect deficiencies or to support the partitioning of functions for the electric and electronic design.
- Separate between system functions and context functions. Context functions are functions in the context of the software under development. They belong to other software parts of the system, can be used, but not changed.

Solution Idea. To address all requirements, a model for the functional design, consisting of three diagram types, was developed. As it is common to use complementary diagrams to describe function behavior

(e.g. (Klein et al., 2004)), we suggested the use of interface automata (cf. (Alfaro and Henzinger, 2001)) to describe the functional behavior of single functions. Interface automata can be composed to describe the overall system behavior. To document the functional hierarchy, we suggested the use of feature trees (cf. (Kang et al., 1998)). On the one hand, these diagrams are common in industry and, on the other hand, they provide the opportunity to describe variability which is of specific relevance to the automotive industry. In addition, to document functional dependencies, we suggested the use of function network diagrams. Current notations lack the necessary formalization to address industry needs regarding automation (cf. (Brinkkemper and Pachidi, 2010)). Therefore, a formalized diagram type was defined, that is compatible to the informal diagrams currently used in industry (e.g. (Jantsch and Sander. 2000), (Grönniger et al., 2008), or (Beeck, 2007)).

3.4 Solution Approach

As explained in Section 2, a solution approach was developed (i) to address the revealed industry needs and thereby improve current function-centered engineering processes, and (ii) to validate and concretize the revealed industry needs themselves. While several solution ideas approved by industry professionals were discussed within Sections 3.1, 3.2, and 3.3, this section summarizes the current state of the overall approach and discusses necessary future enhancements. A more detailed view on several parts of the solution can be found in (Daun et al., 2014).

The main idea is based on the use of dedicated review models to support continuous model evolution between behavioral requirements and functional design, and to support correctness checking of the functional design. The solution approach mainly uses adaptations and enhancements of existing techniques to address the automated creation of the review model and its back transformation. Therefore, model synthesis and transformation were designed to address behavioral requirements (described by use of message sequence charts) and a functional design (described by function network diagrams, function behavior diagrams, and function hierarchy diagrams). The process steps for creation and processing of the review models are sketched in Figure 4. To review and evolve the functional design, the requirements engineer is provided with a behavioral requirements like description of the functional design, the functional architect is provided with a functional design like description of the behavioral requirements.

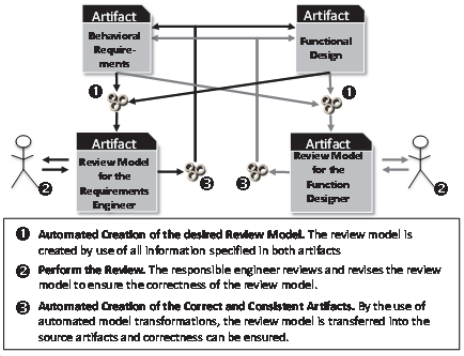


Figure 4: Proposed solution concept.

As discussed, future work will have to deal with the automated analysis of the functional design to support the creation of subsequent artifacts like the deployment architecture. Promising techniques to foster this issue were identified, but were not applied to the case studies, yet. Such automated techniques are also desired to support the review itself, for example to determine, which parts of the review model are probably incorrect and what solutions could be like. In addition, the current solution revealed the desire for more concrete diagrams within the review models. For example, to detect functional interplay and to decide, whether this interplay is desired or not specific diagrams, detailing only the interplay could support the engineers. Therefore, current approaches to detect implied scenarios (e.g. (Letier et al., 2005), or (Alur et al. 2000)) and approaches dealing with the feature interaction problem e.g. ((Felty and Namjoshi, 2003), or (Shiri et al., 2007)) seem promising.

4 DISCUSSION & CONCLUSION

We presented a study conducted to gain insights into industry needs regarding the function-centered engineering of embedded systems. The study design, the major industry needs revealed and ideas for suitable solutions to the industry needs were discussed in more detail. It revealed that industry desires automated method support to foster continuous engineering and model evolution as well as correctness checking of the functional design, which is the main artifact in function-centered engineering. As fully automated techniques were evaluated as not applicable in every situation, the presented preliminary solution approach is widely based on dedicated review models. These models can be created in automated manner and the review results can be pro-

cessed fully automated as well, but the review itself is mainly based on manual investigations.

As the study was conducted among German industry and the number of participants was small it must be discussed, whether the results are generalizable or not. The participating companies were asked to send only professionals that are representative for the company. Furthermore, the questionnaire was not used to determine industry needs presented within this paper. The questionnaire was used to gain first insights into potential industry needs. Therefore, the results of the questionnaires were qualified by interviews. Afterwards, they were concretized and confirmed by the use of workshops and by the use of industrial case studies. By the combination of these techniques, we assume that the results are representative for at least large European companies in the automotive and avionics domain. Participants stem from worldwide operating companies with branches in other countries, especially throughout Europe, which indicates, that our findings are not bound to national borders. Beside of this, we assume that our findings are probably not valid for all parts of the embedded systems domain: Since the usage of model-based engineering and the desire for consistency and correctness within specifications is related to the development of safety-critical systems.

ACKNOWLEDGEMENTS

The research serving as basis for this paper has has been funded in part by the *German Federal Ministry of Education and Research* (support code: 01IS12005C+M). We thank Carsten Albers (with Inchron), Arnaud Boyer (with Airbus) and Matthias Büker (with the offis institute) for their fruitful support in designing the questionnaire and conducting the study. We thank Nelufar Ulfat-Bunyadi (with paluno) for the inspiring discussions about the paper.

REFERENCES

Alfaro, L. & Henzinger, T. (2001), Interface Automata. *Proc. of the ESEC/FSE*, pp. 109-120.

Alur, R., Etessami, K., & Yannakakis, M. (2000), Inference of Message Sequence Charts. *TSE,* pp. 623-633.

Beeck, M. (2007), Development of logical and technical architectures for automotive systems. *Software Systems Modeling*, pp. 204-219.

Beyer, D., Noack, A. & Lewerentz, C. (2005), Efficient Relational Calculation for Software Analysis. *TSE*, pp. 137-149.

Boehm, B. & Basili, v. (2001), Software Defect Reduction Top 10 List. *IEEE Computer*, pp. 135-137.

Borges, R., Garcez, A. & Lamb, L. (2010), Integrating Model Verification and Self-Adaptation. *Proc. of the ASE*, pp. 317-320.

Brinkkemper, S. & Pachidi, S. (2010), Functional Architecture Modeling for the Software Product Industry. *Proc. of the ECSA*, pp. 198-213.

Broy, M. (2006),Challenges in automotive software engineering. *Proc. of the ICSE*, pp. 33-42.

Clarke, E., Emerson, E. & Sifakis, J. (2009), Model checking: algorthmic verification and debugging. *Commun. ACM*, pp. 74-84.

Cox, L., Delugach, H. & Skipper, D. (2001), Dependency Analysis Using Conceptual Graphs. *Proc. of the ICCS*, pp. 117-130.

Daun, M., Weyer, T. & Pohl, K. (2014), Validating the Functional Design of Embedded Systems against Stakeholder Intentions. *Proc. of the MODELSWARD*, pp. 333-339.

Felty, A. & Namjoshi, K. (2003), Feature Specification and Automated Conflict Detection. *TOSEM*, pp. 3-27.

Giese, H. & Wagner, R. (2006), Incremental Model Synchronization with Triple Graph Grammars. *Proc. of the MoDELS*, pp. 543-557.

Gilb, T. & Graham, D. (1993), Software Inspection. *Addison-Wesley*.

Grönninger, H., Hartmann, J., Krahn, H., Kriebel, S., Rothhardt, L. & Rumpe, B. (2008), View-Centric Modeling of Automotive Logical Architectures. *Proc. of MBEES*, pp. 3-12.

Gross, D. & Yu, E. (2001), From Non-Functional Requirements to Design through Patterns. *Requirements Engineering Journal*, pp. 18-36.

Hélouët, L., & Maigat, P. (2001), Decomposition of Message Sequence Charts. *SDL Forum*, pp. 348-364.

Hermann, H., Ehrig, F., Orejas, K., Czarnecki, Z., Xiong, Y., Gottmann, S. & Engel, T. (2013), Model Synchronization based on triple graph grammars: correctness, completeness, invertibility. *Journal on Software Systems Modeling*, pp. 348-364.

Holzmann, G. (1997), The Model Checker SPIN. *TSE*, pp. 279-295.

ITU. (2011), *Recommendation Z.120.*

Jantsch, A., & Sander, I. (2000), On the Roles of Functions and Objects in System Specification. *Proc. of the 8th Intl. Workshop on HW/SW Codesign*, pp. 8-12.

Kang, K., Kim, S., Lee, J., Kim, K., Kim, G., & Shin, E. (1998), FORM: A feature-oriented reuse method with domain specific reference architectures. *Annals of Softw. Eng.,* pp. 143-168.

Klein, T., Conrad, M., Fey, I. & Grochtmann, M. (2004), Modellbasierte Entwicklung eingebetteter Fahrzeugsoftware bei DaimlerChrysler. *Proc. of Modellierung*, pp. 31-41.

Letier, E., Kramer, J., Magee, J., & Uchitel, S. (2005),

Monitoring and Control in Scenario-Based Require-Requirements Analysis. *Proc. of the ICSE*, pp. 382-391.

Mauw, S., & Reniers, M. (1999), Operational Semantics for MSC'96. *Journal of Computer Networks*, pp. 1785–1799.

Milicev, D. (2002), Automatic Model Transformations Using Extended UML Object Diagrams in Modeling Environments. *TSE*, pp. 413-431.

Pretschner, A., Broy, M., Kruger, I. & Stauner, T. (2007), Software Engineering for Automotive Systems: A Roadmap. *Proc. of FOSE*, pp. 55-71.

Shiri, M., Hassine, J., & Rilling, J. (2007), Feature Interaction Analysis: A Maintenance Perspective. *Proc. of the ASE*, pp. 437-440.

Shull, F., Basili, V., Zelkowitz, M., Boehm, B., Brown, A., Port, D., Rus, I. & Tesoreiro, R. (2002), What we have learned about fighting defects. *Proc. of the Intl. Conf. on SW Metrics*, pp. 133-154.

Weber, M. & Weisbrod, J. (2002), Requirements Engineering in Automotive Development - Experiences and Challenges. *Proc. of the RE*, pp. 313-340.