Using dedicated Review Diagrams to detect Defective Functional Interplay in Function-Centered Engineering

Marian Daun, Andrea Salmon, Thorsten Weyer

paluno – The Ruhr Institute for Software Technology
University of Duisburg-Essen
Gerlingstraße 16, 45127 Essen, Germany
{marian.daun, andrea.salmon, thorsten.weyer}@paluno.uni-due.de

Abstract: Today's embedded systems are highly integrated in their context. Their functional behavior does not only result from individual functions but also arises in the interplay between these functions. Automotive and avionics systems consist of control circuits to determine values in the context and to influence context properties. Thereby, indirect interplay between functions emerges from overlapping control circuits, which influence the same context measurement. Often the resulting interplay must be considered as unintended. It is of importance to detect such unintended functional interplay and to determine whether it is desired but unspecified interplay or defective interplay. To foster detecting defective functional interplay this paper suggests the explicit model-based documentation of context measurements in the functional design. Based on this information, we propose the automated generation of dedicated review diagrams to aid in deciding whether unspecified functional interplay is defective.

1 Introduction

Function-centered engineering is commonly used in the development of automotive and avionics systems to address the challenges resulting from the steadily increasing number of system functions and their complexity (cf. [Br09], [DWP14], [Pr07]). In function-centered engineering processes, the functions of a system and their dependencies are the main point of reference [Da13]. Therefore, architecture design decisions and deployment are influenced to a considerable degree by the dependencies on a functional level.

In addition, functions are explicitly designed as central concept for systematic reuse. In industrial practice, the portfolio of the functions developed by a company is stored in proprietary function libraries. The functional design of systems under development is then composed by selecting appropriate functions from the library. Figure 1 sketches the relation between a system's functional design and a company's function library. The initial functional design of an embedded system is often generated from existing functions that are seen as the best fit. Subsequently, functions are enhanced and adopted to fit the specific purposes of the particular system under development.

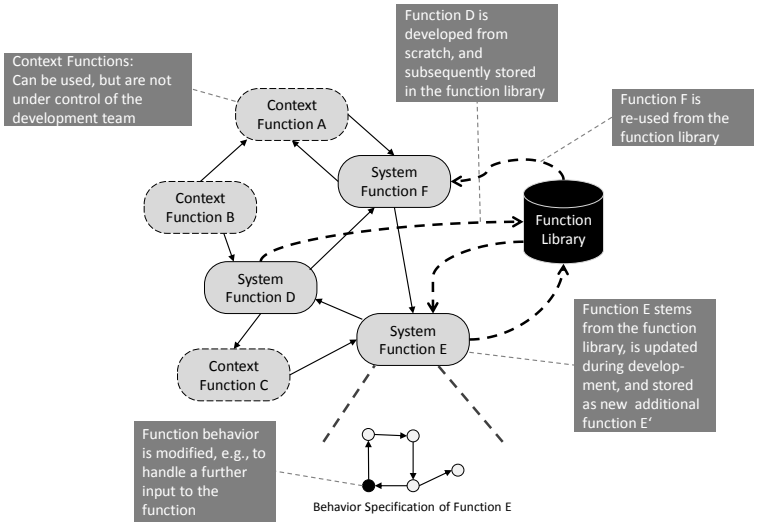

Figure 1: Functional design and function library

By re-using existing functions in different configurations, the functional design commonly provides more functionality than originally intended by the requirements specification. This additional functionality might stem from the use of a single advanced function, or from the interplay between several functions. Especially the latter may remain undiscovered and result in severe threats to the system's safety.

While it is of importance to detect unintended functional interplay, it cannot be detected by simple comparisons with the requirements specification: Since automotive software engineering is largely based on domain knowledge, best practices, and developers' experience, in practice, not all functional interplay that has not been specified explicitly can be assumed to be undesired and defective. By investigations of case examples and expert interviews we gained the insight that a significant amount of defective functional interplay stems from overlapping control circuits, which monitor and control the same context measurement (cf. [DHW14]). Since the context measurements relevant to an automotive system, are well-known at development time, this paper suggests the explicit model-based documentation of context measurements within the functional design.

Figure 2 shows a simplified cut-out from a car's functional design, different control circuits are described by functions and the context measurement that is controlled. The figure exemplifies the interference between different control circuits and how safety defects depend on unrecognized impacts on control circuits. In detail, the figure shows three control circuits, which are in so far overlapping as all three influence the same context measurement: the *torque* generated by the car's engine. In this situation the torque is measured to determine whether the driver accelerates. Based on this evaluation the e-brakes shall be disengaged. In combination with the air-conditioning system, this leads to undesired functional interplay because also the cooling of the car will result in requesting additional torque. If the air-conditioning system increases engine torque in order to cool down the car more, this could result in undesired and unsafe effect on the brakes, which are inadvertently disengaged.

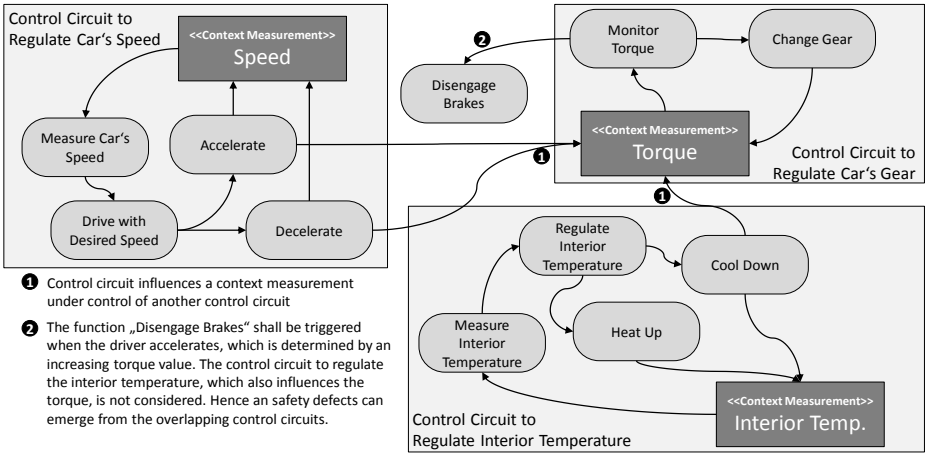

Figure 2: Functions and context measurements build overlapping control circuits

To foster the prevention of defective functional interplay this paper suggests a twofold approach. Since the explicit documentation of implicit knowledge can significantly aid the engineering of complex and safety-critical systems (cf. [Da14]) we suggest the explicit model-based documentation of context measurements in the functional design. The context measurements, their impact to system functions, and the functions influencing the context measurement are documented. This aids, for example, the identification of circular dependencies from context measurements. Since the use of dedicated review models to validate behavioral properties of requirements and functional design is an appropriate way to resolve deficiencies resulting from implicit changes in the stakeholder intentions (cf. [DWP15]), we suggest in the second step the automated generation of dedicated review diagrams, which visualize existing unintended functional interplay. This aids decision making whether the interplay must be considered as defective.

The remainder of this paper is structured as follows: Section 2 reviews the current state of the art on detection of functional interplay. Subsequently, Section 3 presents our solution concept, consisting of the explicit documentation of context measurement and of the automated generation of dedicated review diagrams for analysis purposes. Section 4 reports on first industrial evaluation activity. Finally, Section 5 concludes the paper.

2 Related Work

Several approaches exist to detect emergent properties, which are properties that arise from a specification, but are not part of the specification itself. The detection and the prevention of these properties are discussed in function specifications by means of feature interactions and in behavioral specifications by means of implied scenarios. Furthermore, common approaches to determine and prevent safety hazards, like the failure mode and effects analysis (FMEA), are designed to detect any kind of undesired interplay. However, safety analyses in this sense do not explicitly take into account context measurements and are mostly conducted after the system has been specified (i.e. based

on platform-specific models or code). In contrast, explicit documentation of context measurements support the engineering in earlier stages of development. In addition, manifold approaches exist to detect dependencies between functions, instances, or entities, which are mostly applicable during requirements engineering or functional design phases.

Approaches dealing with the feature interaction problem (e.g., [FN03], [KB98], or [SHR07]) stem from the telecommunication domain. They have partly been transferred to the automotive domain as well (cf. [Ju08]). The detection and prevention of feature interactions is defined as change-request problem: an existing correct specification is altered by a single feature. This feature does not only add its own behavior, but results in undesired behavior in combination with other features. It is to note that in this case a feature is comparable to a system function. As the functional design is developed from a wide variety of different newly developed, changed, and re-used functions, approaches dealing with feature interactions are not applicable, as they focus on one single change. Furthermore, according to the definition of feature interactions, any kind of resulting interplay between functions is undesired. In contrast, the interplay of functions is often explicitly introduced in the functional design and hence desired.

Approaches dealing with implied scenarios (e.g., [AEY00], [Le05], or [UKM01]) aim at detecting system behavior, which emerges from an interaction-based behavior specification, but has not been specified explicitly in a single diagram. Commonly, model-synthesis techniques are used to derive an overall model from partial diagrams. Subsequently, properties of the derived model can be examined and classified by experts. Undesired properties must be removed from the specification and desired properties will be documented explicitly on their own. Current techniques investigate implicit behavior in sequence diagrams and must be adopted to fit the functional design. In addition, current approaches do not take context measurements and resulting implicit changes to overlapping control circuits into account.

Several techniques to determine dependencies between different model elements exist (e.g., [Cl07], [MGP09], or [Sp07]). Especially approaches dealing with the automated detection and documentation of traceability information deal with several kinds of dependencies. These techniques do not consider context measurements to evaluate the existence of dependencies and can thereof not be used to detect functional interplay resulting from overlapping control circuits.

3 Solution Concept

This section introduces our solution concept to detect functional interplay, which emerges from the manipulation of context measurements by multiple functions or systems. The solution idea relies on the model-based documentation of the context measurements and their interdependencies (see Section 3.1), which is commonly seen as a promising solution in the embedded industry (cf. [STP11]). Subsequently, the documented information serves as an input for automated analysis techniques (see Section 3.2).

3.1 Explicit Documentation of Context Measurements in the Functional Design

In the development of today's automotive systems the functional design serves as core development artifact (cf. [Br09]). The functional design defines the functionality of the system under development by means of logical functions. The subsequent development phases rely on this functional design, e.g., to partition the functions into logical components, to develop an adequate electric and electronic design, or to deploy the system functions onto the control units. Therefore, the functional design typically defines the system functions to be developed, functions that are not part of the system under development, but are used by the system under development (in the following: context functions), and the interplay between the aforementioned functions.

The functional design defines structural properties, like the functions' hierarchical relations and the possible interactions, which are exchanged between the functions, and behavioral properties, to specify single functions behavior and to specify the behavior resulting from the interplay of multiple functions. Different diagram types can be used to define a proper functional design under consideration of all relevant perspectives:

- Function hierarchy diagrams define the hierarchical structure of the system functions. Therefore, feature trees can be used [Ka98].
- Function network diagrams define the interactions exchanged between all involved functions from a structural perspective. Function network diagrams are much akin to [JS00]. Figure 3 depicts an excerpt of the functional design of a lane keeping support as an example for a function network diagram.
- Function behavior diagrams define each function's behavior. Therefore, interface automata may be used [AH01].

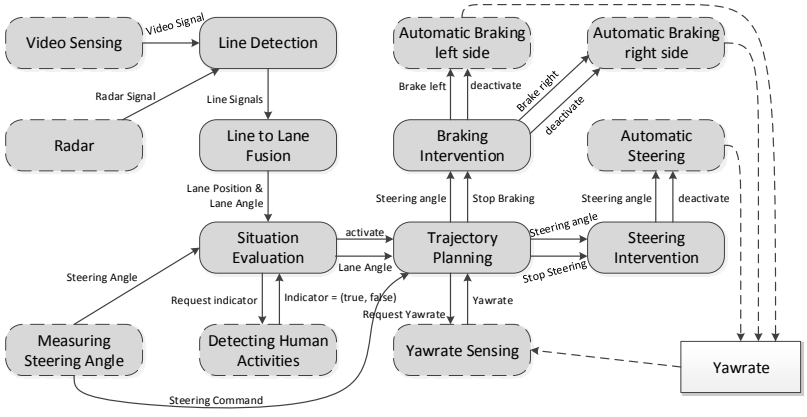

Figure 3: Function network diagram of a lane keeping support

Undesired functional interplay often emerges from interactions with the system's context in which the system's functions react on monitored context information. To unveil possible undesired interplay resulting from control circuits we suggest the explicit documentation of the involved context measurements in the functional design. This enables the engineer to identify other control circuits that might conflict the desired functionality. In addition, this allows for automated support to detect and display functional interplay as

outlined in Section 3.2. Figure 4 depicts another excerpt from the lane keeping support: the function behavior diagrams from the functions 'Trajectory Planning' and 'Steering Intervention', which were already shown in Figure 3. As can be seen, the explicit documentation of context measurements allows for graphical documentation of interrelated function behavior that stems from changing context measurements. In detail, the *yawrate* sensed by the function 'Trajectory Planning' is influenced by the function 'Steering Intervention'.

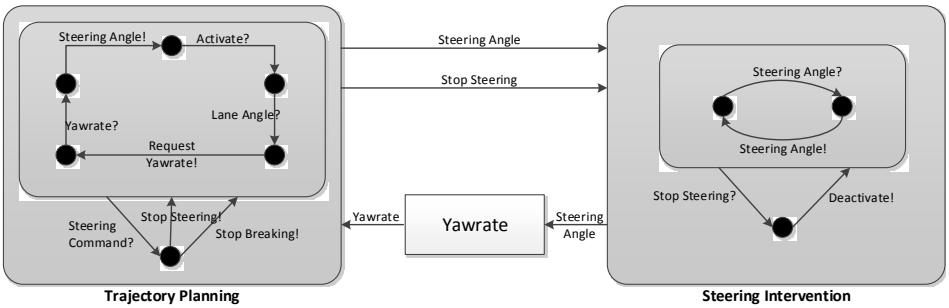

Figure 4: Function behavior diagrams with explicit interrelation through context measurements

As the state of the art lacks formalization of the functional design to aid automated techniques (cf. [BP10]), we give insight in a formalization of the functional design, which explicitly considers context measurements.

The overall model of the functional design can be described as 7-tupel $(F^S, F^C, I, M, \alpha, \beta, \gamma)$. The functional design consists of the following elements:

- A set of functions $F = F^S \cup F^C$. Where F^S is the set of system functions under development and F^C is the set of context functions under consideration.
- A set of interactions I, which are exchanged between the functions.
- A set of context measurements M, which are monitored, or influenced by at least one function.
- A relation $\alpha \subseteq F \times I \times F$, which defines the interactions exchanged between two functions.
- A relation $\beta \subseteq F \times F$, which defines the set of sub functions of one function.
- A relation $\gamma = \gamma^C \cup \gamma^S$, where $\gamma^C \subseteq F \times M$ defines, which context measurement is influenced by a function and $\gamma^S \subseteq M \times F$, which defines if a function is influenced by a context measurement.

By enhancing the work of [AH01] by context measurements, the behavior of each function f of the functional design can be defined as 8-tupel $(V_f, V_f^{init}, A_f^I, A_f^O, A_f^H, M_f, \tau_f, \upsilon_f)$. The function behavior description consists of the following elements:

- Internal states: V_f is the set of all states and V_f^{init}, the set of initial states.
- A set of Actions $A_f = A_f^I \cup A_f^O \cup A_f^H$. A_f^I is the set of inputs, A_f^O is the set of outputs, and A_f^H a set of internal actions.
- A set of context measurements M_f, which are monitored or influenced.

- A transition relation connecting actions and states: $\tau_f \subseteq V_f \times A_f \times V_f$
- And a relation, which defines the context measurements that are affected by a message: $\upsilon_f \subseteq A_f \times M_f$

To ensure consistency between the overall structure of the functional design and its single behavior descriptions, well-formedness rules have to be fulfilled between functions and interactions, such that:

$$\forall\, (f_O, i, f_I) \in \beta\colon f_O, f_I \in F, i \in I \wedge i \in (A^O_{f_O} \wedge A^I_{f_I})$$

Also well-formedness rules are defined for the influence onto context measurements:

$$\forall\, (f, m) \in \gamma^C\colon \exists (a, m) \in \upsilon_f \wedge a \in (A^O_f)$$

$$\forall\, (m, f) \in \gamma^S\colon \exists (a, m) \in \upsilon_f \wedge a \in (A^I_f)$$

3.2 Generating dedicated Review Diagrams to aid Analysis of Functional Interplay

While the explicit documentation of context measurements allows for visual inspections of the functional design to detect undesired functional interplay, in practice, automated support is necessary to keep track with complex and large specifications (cf. [DHW14]). Therefore, it is desirable to present functional interplay in single review diagrams that aid deciding whether a behavioral property resulting from functional interplay is desired. Message sequence charts [ITU11] have proven useful for formal verification due to their formal semantics and yet intuitive graphical notation commonly used during automotive development [WW02]. Figure 5 depicts a basic message sequence chart that displays the undesired functional interplay from Figure 2. It is clearly depicted, that the context measurement torque is affected by the cooling of the car's interior. This effect to the torque leads to disengaging the car's brakes. By visual inspection the engineers detect that this effect is undesired. To resolve this effect, the engineers will, for example, decide to determine the intention of the driver to speed up, not by monitoring the torque but by monitoring the gas pedal position.

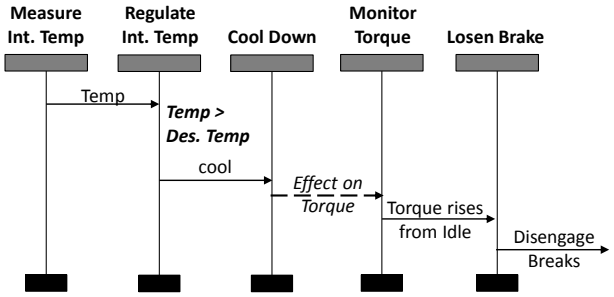

Figure 5: Defective functional interplay is visualized by a message sequence chart

As the functional design consists of a formalized notation, formal methods can be applied to generate the review diagrams in fully automated manner. For example, existing

techniques for path generation based on traceability information can be adopted (e.g., [He04]). And synthesis techniques from the related work (e.g., [AEY00], [Le05], [UKM01]) can be used to display these paths as bMSCs.

4 Industrial Evaluation

We conducted an initial study to determine the state of practice regarding the function-oriented engineering of embedded systems and current challenges (cf. [DHW14]. The study was conducted in the years 2012 and 2013 with several industrial and academic partners. Most partners stem from the automotive domain and may considered as large, German, world-wide operating original equipment manufacturers and suppliers. The study was of qualitative nature with minor quantitative parts. A comprehensive research roadmap was developed for the purpose of the study, which incorporated expert workshops and case studies. Industry partners supported case examples to ensure the appropriateness of the case studies. Excerpts of the functional design of one case example are used in Figures 3 and 4.

During investigation, the need to deal with functional interplay was determined. This seems not only to be valid for the automotive domain, but also for the avionics domain. Investigation showed that functional interplay arising from context measurements, which impact different control circuits, is seen as a major threat to system's safety by industry professionals. Beside this, further kinds of functional interplay exist, which are also needed to be investigated, and are not addressed within this approach (e.g., the unused functionality of re-used functions, may evolve an undesired system behavior in combination with other functions).

Furthermore, it has to be recognized, that the documentation of context measurements is based on the assumption that all relevant context measurements are known. For example, the development team of the air condition may not recognize the increasing throttle value. Industry professionals assured that this issue can be neglected in the current situation, as all context measurements are well-known even before development starts. This can be explained with the experience-oriented development methodology and the already existing huge engineering knowledge. In the current situation development teams can easily check, whether one of these well-known measurements is influenced. Therefore, checklists may be used.

In the interviews it was also recognized that in the development of future systems not all relevant context measurements will be known at design time. As interconnectivity and context-sensitivity rises (this is partly already valid for avionics systems) the interplay with other systems (e.g., planes, cars, or intelligent traffic signs) leads to several effects. First, the control circuits enlarge and the system under development depends on control circuits from other systems. Second, the control circuit may not be under control of only one system, which means that context measurements may be altered by different control circuits of different systems. Third, other systems and other control circuits may be unknown at design time and the system will have to handle effects during runtime. In addi-

tion, systems with long life cycles may be confronted with context measurements, which were not intended during system development.

5 Discussion and Conclusion

In summary, this paper presented an approach to detect undesired functional interplay, which arises from overlapping control circuits in automotive and avionics systems. The approach suggests the explicit documentation of context measurements in the functional design, which is the central artifact during function-centered engineering of embedded systems. Based on the explicit documentation in a formalized functional design, automated model-transformations can be used to detect and to display functional interplay in dedicated review diagrams.

First evaluation result from expert investigations and industrial case studies show the appropriateness of the presented approach for industrial purposes. While this approach seems pretty useful for traditional embedded systems several challenges arise for the engineering of cyber physical systems (cf. [BCG12]). Due to their heterogeneous nature as well as their growing complexity [Fo12], it is, for example, a challenging task to ensure the correctness of such collaborative systems. For example, the types of functions and systems involved in a collaborative system network may be unknown, and additionally, the numbers of participating functions and systems may be uncertain, as well. Hence, future work will particularly have to deal with emergent behavior in the interplay of collaborative system networks under consideration of uncertainty and open contexts.

Acknowledgements

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

References

[AH01] Alfaro, L.; Henzinger, T.: Interface Automata. In: Proc. ESEC/FSE, 2001; pp. 109–120.

[AEY00] Alur, R.; Etessami, K.; Yannakakis, M.: Inference of Message Sequence Charts. In: Proc. ICSE, 2000; pp. 304-313.

[BP10] Brinkkemper, S.; Pachidi, S.: Functional Architecture Modeling for the Software Product Industry. In: Proc. Europ. Conf. Softw. Arch., 2010; pp. 198-213.

[BCG12] Broy, M.; Cengarle, M.V.; Geisberger, E.: Cyber Physical Systems: Imminent Challenges. In: Proc. Monterey WS Large Scale Complex IT Systems, 2012; pp. 1-28.

[Br09] Broy, M.; Gleirscher, M.; Merenda, S.; Wild, D.; Kluge, P.; Krenzer, W.: Toward a Holistic and Standardized Automotive Architecture Description. In: IEEE Computer 42(12), 2009; pp. 98-101.

[Cl07] Cleland-Huang, J.; Settimi, R.; Romanova, E.; Berenbach, B.; Clark, S.: Best Practices for Automated Traceability. In: IEEE Computer 40(6), 2007; pp. 27-35.

[Da13] Daun, M.; Brings, J.; Höfflinger, J.; Weyer, T.: Funktionsgetriebene Entwicklung software-intensiver eingebetteter Systeme in der Automobilindustrie – Stand der Wissenschaft und Forschungsfragestellungen. In: Proc. Envision'13, 2013; pp. 293-302.

[Da14] Daun, M.; Brings, J.; Tenbergen, B.; Weyer, T.: On the Model-based Documentation of Knowledge Sources in the Engineering of Embedded Systems. In: Proc. Envision'14, 2014; pp 67-76.

[DHP14] Daun, M.; Höfflinger, J.; Weyer, T.: Function-centered Engineering of Embedded Systems – Evaluating Industry Needs and Possible Solutions. Proc. Int. Conf. Eval. of Novel Approaches to Softw. Eng., 2014; pp. 226-234.

[DWP14] Daun, M.; Weyer, T.; Pohl, K.: Validating the Functional Design of Embedded Systems Against Stakeholder Intentions. In: Proc. Int. Conf. Model-Driven Eng. and Softw. Dev., 2014; pp. 333-339.

[DWP15] Daun, M.; Weyer, T.; Pohl, K.: Detecting and Correcting Outdated Requirements in Function-Centered Engineering of Embedded Systems. In: Proc. Int. Working Conf. Req. Eng.: Foundations for Softw. Qual., 2015; pp. 65-80.

[Er05] Ericson, C.A.: Hazard Analysis Techniques for System Safety, Wiley, 2005.

[FN03] Felty, A.; Namjoshi, K.: Feature Specification and Automated Conflict Detection. In: ACM Trans. on Softw. Eng. and Methodology 12(1), 2003; pp. 3-27.

[Fo12] Fouquet, F.; Morin, B.; Fleurey, F.; Barais, O.; Plouzeau, N.; Jezequel, J.: A Dynamic Component Model for Cyber Physical Systems. In: Proc. ACM SIGSOFT Symp. on Component Based Softw. Eng., 2012; pp. 135-144.

[He04] Hessel, A.; Larsen, K.; Nielsen, B.; Pettersson, P.; Skou, A.: Time-Optimal Real-Time Test Case Generation Using UPPAAL. In: Proc. Int. WS Formal Approaches to Software Testing, 2004; pp. 114-130.

[ITU11] International Telecommunication Union: Recommendation Z.120. Int. Standard, 2011.

[JS00] Jantsch, A.; Sander, I.: On the Roles of Functions and Objects in System Specification. In: Proc. Int. WS Hardware/Software Codesign, 2000; pp. 8-12.

[Ju08] Juarez Dominguez, A.: Feature Interaction Detection in the Automotive Domain. In: Proc. Int. Conf. Automated Softw. Eng., 2008; pp. 521-524.

[Ka98] Kang, K.; Kim, S.; Lee, J.; Kim, K.; Shin, E.; Huh, M.: FORM: A Feature-Oriented Reuse Method with Domain Specific Reference Architectures. In: Annals of Softw. Eng. 5(1), 1998; pp. 143-168.

[KB98] Kimbler, K.; Bouma, L.: Feature Interactions in Telecommunication and Software Systems V, IOS Press, 1998.

[Le05] Letier, E.; Kramer, J.; Magee, J.; Uchitel, S.: Monitoring and Control in Scenario-Based Requirements Analysis. In: Proc. ICSE, 2005; pp. 382-391.

[MGP09] Mäder, P.; Gotel, O.; Philippow, I.: Enabling Automated Traceability Maintenance through the Upkeep of Traceability Relations. In: Proc. Europ. Conf. Model Driven Architecture – Foundations and Applications, 2009; pp. 174-189.

[Pr07] Pretschner, A.; Broy, M.; Kruger, I.; Stauner, T.: Software Engineering for Automotive Systems: A Roadmap. In: Proc. Int. WS Future of Softw. Eng., 2007; pp. 55-71.

[SHR07] Shiri, M.; Hassine, J.; Rilling, J.: Feature Interaction Analysis: A Maintenance Perspective. In: Proc. Int. Conf. Automated Softw. Eng., 2007; pp. 437-440.

[Sp07] Spanoudakis, G.; Zisman, A.; Pérez-Miñana, E.; Krause, P.: Rule-based generation of requirements traceability relations. In: J. Syst. Softw. 72(2), 2007; S. 105-127.

[STP11] Sikora, E.; Tenbergen, B.; Pohl, K.: Requirements engineering for embedded systems: an investigation of industry needs. In: Proc. Int. Working Conf. on Req. Eng.: Foundation for Softw. Qual., 2011; pp. 151-165.

[UKM01] Uchitel, S.; Kramer, J.; Magee, J.: Detecting Implied Scenarios in Message Sequence Chart Specifications. In: Proc. ESEC/FSE, 2001; pp. 74-82.

[WW02] Weber, M.; Weisbrod, J.: Requirements Engineering in Automotive Development - Experiences and Challenges. In: IEEE Software 20(1), 2002, pp. 313-340.