

SPES XT Modeling Framework

As embedded systems evolve into more and more complex structures to meet the continuously increasing complexity of requirements, they face a variety of challenges. In particular, the involvement of multiple engineering disciplines targeting cross-cutting aspects of the system under development makes the situation even more challenging. Hence, there is a great need to establish a seamless modeling framework that on the one hand, facilitates reuse and automation, while on the other hand, is independent of any application domain. The modeling framework must provide appropriate models and description techniques for modeling the different aspects and artifacts of system development as well as methods and process techniques for creating and analyzing such artifacts. Therefore, this chapter introduces the SPES XT modeling framework, which aims to address these issues.

3.1 Introduction

The SPES modeling framework

From 2010 to 2012, the SPES modeling framework [Broy et al. 2012] was developed in a close collaboration between academia and industry. The SPES modeling framework, which is a structured collection of modeling concepts, enables the seamless model-based engineering of embedded software and relies on the core principles of *divide and conquer* and *separation of concerns*. The framework allows us to manage the complexity of modern embedded systems during the software engineering process. Furthermore, it allows us to apply formal methods for verification and validation purposes, which in turn, for instance, fulfills the need for safety-critical embedded software to work correctly.

SPES XT extensions

The original SPES modeling framework already emphasized the use of the framework for documenting and analyzing certain quality aspects such as safety (see [Höfig et al. 2012]) and real time (see [Hilbrich et al. 2012]). However, in order to address the specific engineering challenges identified, the SPES modeling framework has been extended in two different directions:

- ❑ *Core methodological extensions* to address additional methodological aspects of a general modeling theory for embedded systems
- ❑ *Specific methodological extensions* to address specific engineering challenges in the engineering of embedded systems

The SPES XT modeling framework comprises the SPES modeling framework and, among others, three *core methodological extensions* to the original SPES modeling framework described in [Broy et al. 2012]:

- ❑ The *SPES XT Process Building Block Framework* allows the definition of customized engineering processes for specific purposes based on the artifacts defined in the SPES XT modeling framework (Section 3.3).
- ❑ The *SPES XT Context Modeling Framework* allows consistent documentation and analysis of properties or assumptions about the context of embedded systems and software (Chapter 4).
- ❑ The *SPES XT Systems Engineering Extensions* allow the SPES XT modeling framework to be applied within the overall systems engineering process for embedded systems including the

different systems engineering disciplines, like electrical engineering, mechanical engineering as well as software engineering (Chapter 5).

The *specific methodological extensions* of the SPES XT modeling framework comprise extensions to the original framework with respect to the following specific challenges in the engineering of embedded systems:

- *Early validation of engineering artifacts (Chapter 6)*
- *Verification of systems in physical contexts (Chapter 7)*
- *System function networks (Chapter 8)*
- *Optimal deployment (Chapter 9)*
- *Modular safety assurance (Chapter 10)*
- *Variant management and reuse (Chapter 11)*

3.2 Structure of the SPES XT Modeling Framework

In the SPES XT modeling framework, a system is described from different *viewpoints* capturing different stakeholder concerns and with varying *degrees of granularity*. A system description from a specific viewpoint and with a specific degree of granularity is called a *view* [ISO/IEC/IEEE 42010].

3.2.1 Viewpoints

The SPES XT modeling framework facilitates the seamless, model-based engineering of embedded software with four core viewpoints. These viewpoints distinguish between the problem space and the solution space and between functional, logical, and technical solution concepts. In doing so, the viewpoints address the concerns of different stakeholders. We consider these viewpoints and the corresponding concerns as core viewpoints common for embedded software across all application domains. The four core viewpoints and their concerns are:

Four core viewpoints

The *requirements viewpoint* supports the requirements engineering process in eliciting, documenting, negotiating, and validating requirements for the system under development (SUD). To distinguish between different types of requirements such as assumptions and constraints, goals, behavioral requirements, and more detailed solution-related requirements, the requirements viewpoint differentiates between four types of models: the context model, the goal

Requirements viewpoint

model, the scenario model, and the solution-oriented requirements model. For details on the requirements engineering viewpoint, see [Daun et al. 2012].

Functional viewpoint

The *functional viewpoint* supports the development of a functional system specification for the SUD. Requirements captured in the requirements viewpoint are structured with respect to user functions, which specify patterns of use from an actor's (i.e. an human user or an external system) point of view. User functions are specified by system functions, which are the functional building blocks that the system is intended to provide. These types of models enable a precise analysis of functional dependencies and feature interactions [Vogelsang and Fuhrmann 2013]. For details on the functional viewpoint, see [Vogelsang et al. 2012].

Logical viewpoint

The *logical viewpoint* supports the solution design for the SUD. In this viewpoint, functional building blocks (captured in the functional viewpoint) are realized by communicating components arranged in a component architecture. In contrast to the functional viewpoint, the models of the logical viewpoint are not structured solely with respect to functionality but rather in terms of architectural design. Here, aspects such as the organizational structure, dependability, maintainability, and reusability also play an important role (e.g., components that implement reliability mechanisms). For details on the logical viewpoint, see [Eder et al. 2012].

Technical viewpoint

The *technical viewpoint* supports the technical implementation of the SUD. Components, which are captured in the logical viewpoint and which describe abstract solution elements, are refined into software modules that are executed on a specific execution platform. The mapping from components in the logical viewpoint to elements of the technical viewpoint is called *deployment* (see also Chapter 9). This viewpoint defines how the software and hardware interact to realize the system goals. For details on the technical viewpoint, see [Weber et al. 2012].

3.2.2 Degree of Granularity

The viewpoints describe the SUD with respect to different concerns. However, these descriptions may vary in their degree of granularity. For complex systems in particular, it is reasonable to start with rather high-level descriptions of requirements, functions, components, and technical devices. Once these high-level descriptions have been created, these views are typically refined and detailed step by

step. Therefore, the SPES XT modeling framework supports views with different degrees of granularity.

To change the degree of granularity for a given view to a higher degree, a low-degree view is decomposed into a number of more detailed system views following the principle of divide and conquer. This step can be performed from a view of any viewpoint (e.g., a function is decomposed into subfunctions or a component is decomposed into subcomponents). As a result, we get a view for each element resulting from the decomposition. These views have a higher degree of granularity and also a smaller scope compared to the original system element from which they are derived. Furthermore, we can look at these detailed elements again from different viewpoints to specify their requirements, functions, logical components, and technical components. We can visualize this procedure with a tree-based representation as shown in Fig. 3-1. The root node of the tree represents the entire SUD (S). For this system, we document the requirements in the requirements viewpoint (R), specify the functions in the functional viewpoint (F), design a component architecture in the logical viewpoint (L), and describe the intended execution platform in the technical viewpoint (T). When decomposing the system into an architecture of communicating components in the logical viewpoint, we increase the degree of granularity. In the figure, the system S is decomposed into three logical components (LC1-3), each building a separate engineering path with a higher degree of granularity and with a smaller scope. For each of these logical components, we again document the requirements (R), specify the functions (F), design a component architecture (L), and describe the intended execution platform (T).

Degree of granularity

In this example, we used the view of the logical viewpoint to change the degree of granularity, i.e., the decomposition of the logical viewpoint model determines the next degree of granularity. In general, we can decide separately for each system element which viewpoint to use to increase the degree of granularity. Automotive manufacturers, for example, usually structure their vehicle systems with respect to functions (functional viewpoint), which are subsequently engineered by suppliers (who may specify component architectures for the functions). System integrators, on the other hand, would usually detail views of the technical viewpoint because they integrate hardware, basic software, and application software.

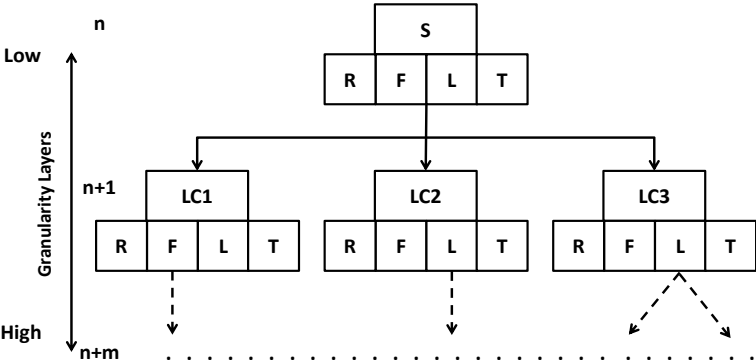


Fig. 3-1 Tree-based representation of granularity layers

Engineering path

The development of views for a system element with a specific degree of granularity is summarized in an *engineering path*. When decomposing a system element into a set of more detailed system elements, the engineering process is split up into an engineering path for each detailed system element. We illustrate this idea of engineering paths in Fig. 3-2, which shows a different representation of the example from Fig. 3-1. In the figure, the engineering path for system *S* is split up into three engineering paths for logical components *LC1-3*, which together define the next granularity layer. Within the engineering paths, again the models of the viewpoints are created for the corresponding system element.

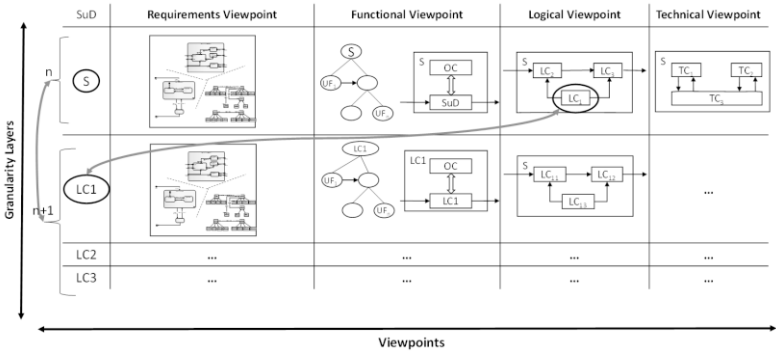


Fig. 3-2 Structuring of the engineering process by granularity layers

Development steps

When considering the models that represent the views in the SPES XT modeling framework, we can explain the progression with decomposition and refinement relations:

- ❑ *Decomposition*: Decomposition describes a model as an ensemble of smaller model elements and their relationships. In

the SPES XT modeling framework, decomposition is used to increase the degree of granularity. Functional decomposition, for example, allows a structuring of the interface behavior of the overall system in terms of functions leading to a function hierarchy. A logical decomposition describes the decomposition of the overall system into subsystems.

- **Refinement:** Refinement describes the transformation of an abstract model into a more concrete model with the same properties as the abstract model. For instance, the refinement relations can address the description of the system interface. Technical signals are a refinement of logical messages, which may be refinements of events used to formulate requirements. The resulting behavior described by the models must be consistent with each other. For example, the logical model must fulfill the behavior of the functional model but may also ensure some quality characteristics (e.g., behavior in the case of failures).

The use of mechanisms for abstraction, decomposition, or the definition of different degrees of granularity is common in the engineering of embedded systems. The SPES XT modeling framework employs these mechanisms to enable systematic engineering of embedded systems. Furthermore, these mechanisms are mapped to specific model types with clear relationships to each other.

*Contributions of the
SPES XT modeling
framework*

3.3 SPES Process Building Block Framework

It is important to distinguish between the artifact structure that is used in system development and the development process itself. On the one hand, the artifact structure defines which contents are worked out and documented and how they are related. On the other hand, the development process defines by which techniques and in which order the artifacts are created. While the SPES XT modeling framework focuses on the artifact structure, the SPES XT Process Building Block Framework [Daun et al. 2016] provides a methodology for documenting processes, which describe the creation and analysis of artifacts. Hence, the SPES XT Process Building Block Framework and the SPES XT modeling framework complement each other and provide a consistent methodology for the engineering of embedded software.

*Defining development
processes*

3.3.1 Overview of the Framework

Building blocks

The relationships between artifacts are specified by relations. Such a relationship can be expressed by a process building block that defines a general technique for artifact creation and analysis. This can be described, for example, as an algorithm, a guideline, or a tool. A building block has an input (e.g., an engineering artifact or stakeholder knowledge) and an output, (e.g., an engineering artifact or an analysis result). Furthermore, preconditions and postconditions can be associated with building blocks to meet custom requirements of applicability for a specific process. There are two main types of process building blocks:

Construction building block

❑ The output of a *construction building block* is always an artifact that belongs to a specific viewpoint. For example, the derivation of a logical architecture from a functional architecture creates an artifact that belongs to the logical viewpoint. A construction building block may or may not have some artifact-related input. For example, a context diagram may be created from scratch based on information that might not even be documented.

Analysis building block

❑ *Analysis building blocks* are used to analyze engineering artifacts. They could be, for example, a consistency check between two artifacts from different viewpoints or from different degrees of granularity. Analysis building blocks have some input and their output is an analysis result, which can be an artifact that cannot be mapped to a specific viewpoint of the SPES XT modeling framework even though it might be a model.

Combining building blocks

Processes can be combined to form larger building blocks. On the one hand, this building block concept provides flexibility as it does not impose a rigid process. On the other hand, it allows building blocks to be reused within the same process as well as in other processes.

Process definition

To define a development process, building blocks are combined: both construction building blocks and analysis building blocks can be used. The output of one building block can be the input of another building block. Furthermore, one building block's postcondition might ensure that another building block's precondition is met. Not only building blocks themselves but also sequences of building blocks can be reused in other processes.

Building block templates

Custom process templates can be defined to characterize common processes in various engineering challenges. For example, the derivation of a functional architecture from a requirements specifi-

cation based on message sequence charts (MSCs) can be defined as a building block template which can be reused in a variety of engineering challenges.

3.3.2 Relationship to the SPES XT Modeling Framework

An important distinction in the SPES XT building block framework is the structuring of the results of the development process, which are artifacts, and the relationships between them. The way artifacts are defined during the development process often varies. However, there are some basic concepts and relationships defined in the SPES XT modeling framework that can be summarized as follows:

- ❑ *Init building blocks*: An artifact that belongs to a viewpoint can be created from scratch, meaning that the corresponding technique does not rely on other artifacts. For example, the initial artifacts of the requirements viewpoint are developed from scratch (see Fig. 3-3 (1)). To do so, the requirements engineer typically consults the stakeholders and elicits their intentions.
- ❑ *Relationships of artifacts between viewpoints*: a construction building block can relate artifacts of viewpoints on the same engineering path. This enables relationships between artifacts of any viewpoint to be modeled on the same degree of granularity (horizontal relation of artifacts, see Fig. 3-3 (2)). This allows, for example, the generation of an initial version of the functional design (i.e., a major engineering artifact in the functional viewpoint) from the existing requirements artifacts.
- ❑ *Relationships of artifacts between different degrees of granularity*: construction building blocks can relate artifacts with the same viewpoint but with different degrees of granularity. By combining the building blocks, we achieve higher-order relationships between artifacts with different degrees of granularity (vertical relation of artifacts among the viewpoints, see Fig. 3-3 (3)).
- ❑ *Relationships between artifacts belonging to the viewpoints and artifacts outside the viewpoints*: in all phases of the development process, there are certain artifacts that are defined but are not related to the SPES XT modeling framework, see Fig. 3-3 (4). For example, the logical architecture is often used to create simulations and tests for the SUD. The results of this simulation and testing phase are not part of the SPES XT modeling framework, although they may be used for the correction of the functional design, which is again an artifact belonging to

the SPES XT modeling framework. Thus, analysis building blocks are very important, not only because they allow the analysis of certain artifacts but also because they provide the ability to trace back the derivation of certain artifacts.

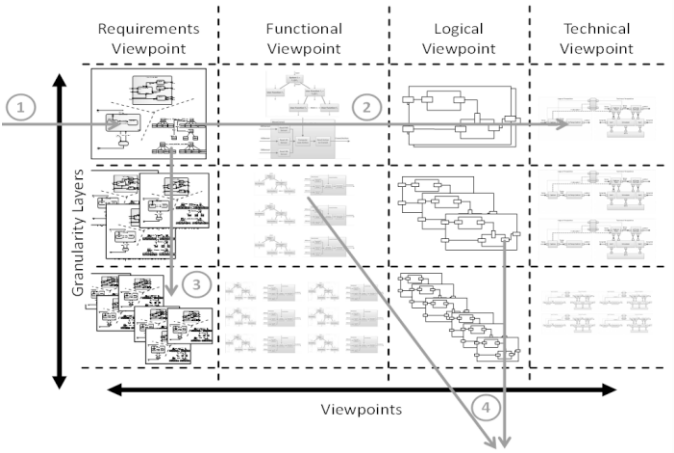


Fig. 3-3 Relationships between artifacts in the SPES XT modeling framework

3.3.3 Example: Beach Well Case Study

The following example depicted in Fig. 3-4 illustrates the use of the SPES XT process building block framework for the beach well case study. It describes the process of creating a formal requirements specification from informal system requirements (creation block C-1), a corresponding functional design (creation block C-2), the creation (analysis block A-3) and use (analysis block A-4) of a verification model, and the correction of the functional design (creation block C-5) and requirements specification (creation block C-6) with the selected degree of granularity.

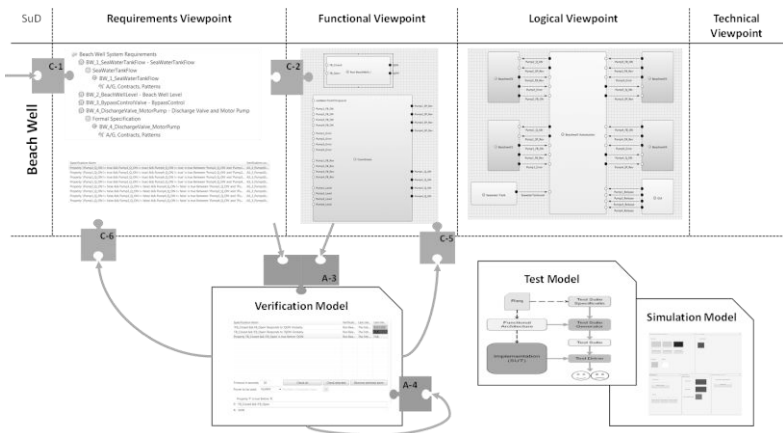


Fig. 3-4 Artifact structure and building blocks for the beach well case study

Fig. 3-5 illustrates the combination of the basic building blocks defined above to describe an exemplary development process for beach wells. A template is filled out for each building block, with the applied methodology explained in more detail. Based on the schema above, custom development processes may be defined for the aspects of interest. For example, in the logical viewpoint, a logical architecture may be derived from a functional architecture. Furthermore, the logical architecture may be used for testing and simulation. These results are then needed again for the correction of the functional design, and so on.

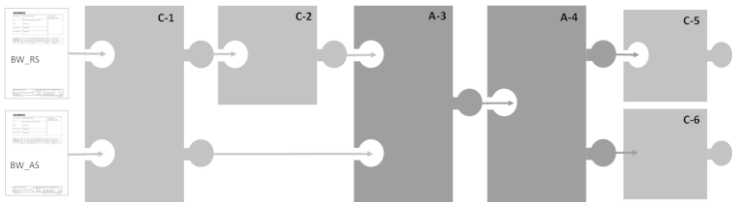


Fig. 3-5 Building blocks of the engineering process for beach wells

3.4 Specific Extensions of the SPES XT Modeling Framework

The extensions of the modeling framework support the need for tailoring and instantiating the modeling framework in different problem domains and for specific problem classes. Thus, the SPES XT modeling framework provides different extension mechanisms

for specific purposes to allow for seamless artifact-based engineering of embedded systems.

Core model

The main strategy for purpose-specific extensions of the SPES XT modeling framework is the definition of metamodel extensions that partly use the core concepts and add further concepts. This technique of extending the SPES XT modeling framework has already been applied to six general engineering challenges in the engineering of embedded systems. However, the SPES XT modeling framework can be extended in a similar way to deal with other problem classes as well. The SPES XT modeling framework core model, consisting of the four core viewpoints and the core views on different degrees of granularity, can be extended by certain elements. For example, the basic concepts of a functional architecture from the core functional viewpoint can be extended to allow more sophisticated functional architectures to consider collaboration between different networked systems.

Core model extensions

Core model extensions are designed to describe individual extensions for each domain. On the one hand, modeling framework extensions that adapt the core metamodel to the needs of a specific domain may be defined. On the other hand, custom building blocks may be useful for modeling individual development processes according to the needs of a specific engineering challenge. In conclusion, each engineering domain defines its own extensions based on the core model to address its specific engineering challenges.

Crosscutting concerns

For some purposes, it is necessary to address crosscutting concerns, which affect each viewpoint, each degree of granularity, and furthermore, other extensions. Such crosscutting concerns might be, for example, real time or variability. Crosscutting extensions are related to qualities of embedded systems affecting all viewpoints. Crosscutting extensions are orthogonal to the viewpoints (see Fig. 3-6). They define ontological elements and relationships that may enhance existing engineering artifacts. Therefore, the ontological entities may be defined specific to viewpoints, specific to degrees of granularity, or even applicable to all viewpoints, or to all degrees of granularity. The ontological elements of the extensions are related to each other, for example, to ontological elements of another viewpoint or another degree of granularity. The crosscutting specific ontological elements must be related to the ontological elements of the core engineering artifacts of the SPES XT modeling framework. Defining further crosscutting extensions means that the artifacts of two crosscutting extensions will probably be related to one another as well. The SPES XT modeling framework in combination

with the SPES XT process building block framework provides the necessary modeling concepts to achieve a unified and consistent method of modeling ontological relationships of crosscutting extensions.-

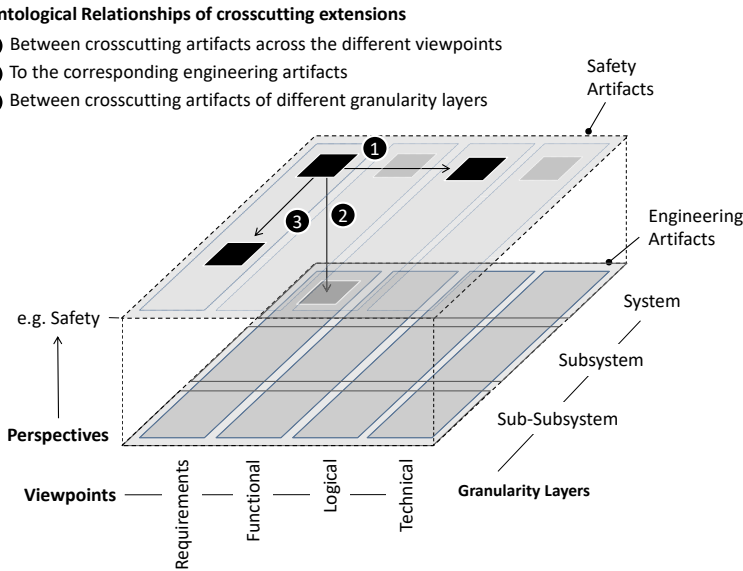


Fig. 3-6 Defining crosscutting extensions in the SPES XT modeling framework (see [Heuer et al. 2013])

3.5 Summary

In this chapter, we presented a conceptual overview of the SPES XT modeling framework. In particular, we illustrated how the framework supports the specification process for embedded software with different degrees of granularity. Furthermore, we introduced the SPES XT Process Building Block Framework, which provides a means for defining engineering processes based on fine-grained methodological building blocks. Hence, the SPES methodology provides the necessary concepts and techniques for defining artifact structures, their relationships and the corresponding engineering steps. Finally, we presented extension mechanisms of the SPES XT modeling framework. These mechanisms can be used to extend the methodological framework to deal with customized engineering problems from different application domains.

3.6 References

- [Broy et al. 2012] M. Broy, W. Damm, S. Henkler, K. Pohl, A. Vogelsang, T. Weyer: Introduction to the SPES Modeling Framework. In: K. Pohl, H. Hönniger, R. Achatz, M. Broy: *Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology*. Springer, Heidelberg/New York, 2012.
- [Daun et al. 2012] M. Daun, B. Tenbergen, T. Weyer: Requirements Engineering Viewpoint. In: K. Pohl, H. Hönniger, R. Achatz, M. Broy: *Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology*. Springer, Heidelberg/New York, 2012.
- [Daun et al. 2016] M. Daun, P. Bohn, J. Brings, T. Weyer: Structured Model-Based Engineering of Long-living Embedded Systems: The SPES Methodological Building Blocks Framework. *Softwaretechnik-Trends*, Vol. 36, No. 1, 2016.
- [Eder et al. 2012] S. Eder, J. Mund, A. Vogelsang: Logical Viewpoint. In: K. Pohl, H. Hönniger, R. Achatz, M. Broy: *Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology*. Springer, Heidelberg/New York, 2012.
- [Heuer et al. 2013] A. Heuer, T. Kaufmann, T. Weyer: Extending an IEEE 42010-Compliant Viewpoint-Based Engineering Framework for Embedded Systems to Support Variant Management. In: *Embedded Systems: Design, Analysis and Verification, IFIP Advances in Information and Communication Technology*, Vol. 403, Springer, Heidelberg 2013, 283-292.
- [Hilbrich et al. 2012] R. Hilbrich, J. van Kampenhout, M. Daun, T. Weyer, D. Sojer: Modeling Quality Aspects: Real-Time. In: K. Pohl, H. Hönniger, R. Achatz, M. Broy: *Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology*. Springer, Heidelberg/New York, 2012.
- [Höfig et al. 2012] K. Höfig, M. Trapp, B. Zimmer, P. Liggesmeyer: Modeling Quality Aspects: Safety. In: K. Pohl, H. Hönniger, R. Achatz, M. Broy: *Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology*. Springer, Heidelberg/New York, 2012.
- [ISO/IEC/IEEE 42010] ISO/IEC/IEEE. Systems and software engineering – Architecture description. ISO/IEC/IEEE 42010:2011(F), International Organization for Standardization, Geneva, Switzerland. 2011.
- [Vogelsang and Fuhrmann 2013] A. Vogelsang, S. Fuhrmann: Why feature dependencies challenge the requirements engineering of automotive systems: An empirical study. In: *Proceedings of the 21st IEEE International Requirements Engineering Conference (RE)*. 2013.
- [Vogelsang et al. 2012] A. Vogelsang, S. Eder, M. Feilkas, D. Ratiu: Functional Viewpoint. In: K. Pohl, H. Hönniger, R. Achatz, M. Broy: *Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology*. Springer, Heidelberg/New York, 2012.
- [Weber et al. 2012] R. Weber, P. Reinkemeier, S. Henkler, I. Stierand: Technical Viewpoint. In: K. Pohl, H. Hönniger, R. Achatz, M. Broy: *Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology*. Springer, Heidelberg/New York, 2012.

Advanced Model-Based Engineering of Embedded
Systems

Extensions of the SPES 2020 Methodology

Pohl, K.; Broy, M.; Daembkes, H.; Hönninger, H. (Eds.)

2016, XII, 303 p. 98 illus., Hardcover

ISBN: 978-3-319-48002-2