Markus Fockel
Peter Heidl
Jens Höfflinger
Harald Hönninger
Jörg Holtmann
Dr. Wilfried Horn
Dr. Jan Meyer
Dr. Matthias Meyer
Jörg Schäuffele

12

Application and Evaluation in the Automotive Domain

This chapter summarizes the application and evaluation of the SPES engineering methodology in the automotive domain. After introducing the particular domain characteristics, we state some research questions that we have investigated. Some of the activities that address these research questions are presented in detail. We conclude that the SPES engineering methodology is a good basis for the development of automotive systems, but could be further refined to fit the particular needs of the domain.

K. Pohl et al. (eds.), *Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology*,
DOI 10.1007/978-3-642-34614-9_12, © Springer-Verlag Berlin Heidelberg 2012

12.1 Overview: Application Domain Automotive

With total revenue of €315 billion, and thus representing approximately 20% of German industrial production, the automotive industry is Germany's most important economic sector, offering employment to 714,000 people. Investing €20 billion in research and development, the automotive industry is the most innovative sector and thus contributes significantly to securing Germany as a location for business (source: VDA annual report 2011).

The necessary innovations are made possible by the enabling technology of embedded software systems. With regard to the total development effort, embedded systems make up between 30 and 40 percent of the value chain with an upward trend.

Across all domains, engineering of embedded systems is characterized by a physical context with real-time requirements and the necessity for interdisciplinary cooperation. Additionally, the automotive domain has a high share of quality requirements, cost pressure, and resource constraints. This is due to high product volumes ranging in the millions, particularly demanding safety and reliability requirements, and extensive variability stemming from a large number of system approaches and functional configurations.

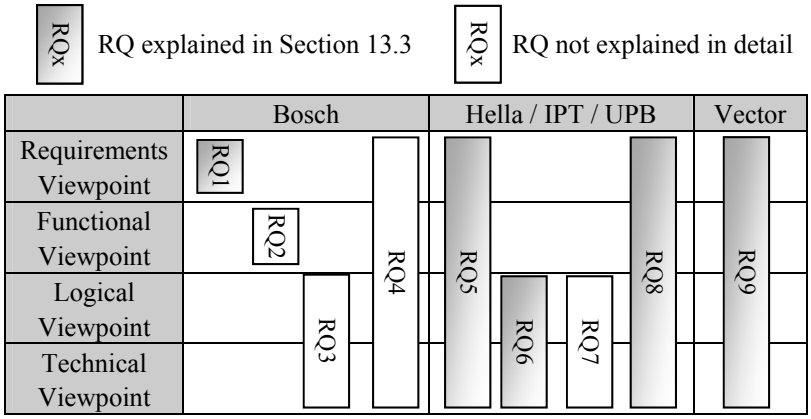
An integrated modeling approach is a key factor to mastering diverse requirements and being able to handle variants in highly complex surroundings. Both of these capabilities are vital in order to meet the challenges of the industry, which is why we worked on developing them in the SPES 2020 project.

12.2 Evaluation Strategy and Correlations to the SPES Modeling Framework

In the automotive application domain, the companies Hella KGaA Hueck & Co., Robert Bosch GmbH, and Vector Informatik GmbH, as well as the academic partners Fraunhofer IPT and University of Paderborn (UPB), worked on a variety of domain-specific *research questions* (RQs). To address the wide range of these questions, various research activities were conducted considering different types of case studies. RQs 1-4 focused on an evaluation of some of the approaches introduced in Part II of this book. RQs 5-8 evaluated and refined the SPES modeling framework (cf. Chapter 3) where necessary in order to exploit domain-specific information and to support domain-specific languages and

standards. RQ9 considered variant handling including tool support covering all viewpoints. The research activities conducted can be related to the viewpoints of the SPES modeling framework as shown in Tab. 12-1.

Tab. 12-1 *Overview of research questions in the automotive domain*



In the following, we will outline each RQ.

- *RQ1:* How can we apply model-based requirements engineering to the automotive domain? Employing the model types of the requirements viewpoint (see Chapter 4), we conducted case studies on an example engine control system and air system. Valuable input for the method developers was gathered from domain experts (see Section 12.3.1).
- *RQ2:* How can we address model-based function development throughout the automotive development life cycle? We analyzed the tool AUTOFOCUS3 by means of a case study conducted on an example engine control system.
- *RQ3:* How can we address safety design in model-based automotive development? The work on this RQ led to the development of the Vertical-Safety-Interface approach [Zimmer et al. 2011]. Industrial feedback was gathered via a case study conducted on a reallocation scenario.
- *RQ4:* How can we empirically validate the methods developed in the automotive domain? The resulting effect on complexity and efficiency was validated empirically in cooperation with Fraunhofer IESE.
- *RQ5:* How can we get from mainly informal requirements to an implementation based on the domain-specific AUTOSAR standard

[AutomotiveSIG 2010] in a more systematic way? A big challenge here was the assurance of consistency and traceability between artifacts of different development phases and viewpoints. A seamless model-based development methodology compliant to Automotive SPICE [AutomotiveSIG 2010] was developed. It addresses this research question using semi-automatic transitions between development phases and viewpoints [Holtmann et al. 2011b].

- *RQ6:* How can we identify design flaws regarding functional correctness and timing in early development phases? To address this question, different kinds of simulation techniques were integrated into the development methodology (cf. RQ5).
- *RQ7:* How can we apply the analysis techniques based on the SPES modeling framework (see Chapter 2) to AUTOSAR architectures? In order to answer this question, a concept for transforming AUTOSAR models into models corresponding to the SPES modeling framework was developed.
- *RQ8:* Is the conceived automotive development methodology (cf. RQ5) applicable and what are its limits? The methodology was evaluated by a proof of concept and developers were asked for feedback as to its feasibility.
- *RQ9:* How can we develop embedded systems for vehicles in a product line approach? Work on this RQ extended the concepts and features of the PREEvision tool to manage product lines and to derive consistent product variants of automotive embedded systems.

Some of these RQs are explained in more detail in the following section.

12.3 Detailed Experience Reports

In total, the RQs we addressed in our activities in the automotive domain cover all constructive development phases for automotive systems. In this section, we present a selection of our activities in more detail. Following the development process, we start with requirements engineering by presenting an evaluation of the requirements view (see Chapter 4) of an engine control system (see Section 12.3.1). We then continue (in Section 12.3.2) with a seamless development methodology from the requirements to an implementation based on AUTOSAR, which was evaluated on a body control module. This ECU offers a wide range of functions related to the car body (e.g., indicator control and interior light control) and communicates with several other ECUs. Finally, we present an approach for handling variants, which is necessary in all steps of the development process (Section 12.3.3).

12.3.1 Requirements Engineering Using the Model of the Requirements Viewpoint

In order to address RQ1 regarding the use of model-based requirements engineering in the automotive domain, we analyzed methods by applying them to the development of software for engine control systems. We analyzed this situation using an air system (system controlling the airflow to the engine) as an example.

Measuring the performance of model-based development methods, including model-based requirements engineering, relies heavily on how adequately models can describe the relevant characteristics of a system regarding both functionality and quality. Customer requests, testability of design and implementation artifacts, and shaping the development process are some of the key reasons for requirements engineering. The evaluation of the model of the requirements viewpoint was motivated by the following questions:

- Does the approach cover relevant functional problem classes?
- Does the approach support the step from the problem domain to the solution domain?
- Does the approach scale to large systems?
- Can the approach address the variability inherent to the domain adequately?
- Does the approach allow statements regarding the completeness or unambiguousness of requirements modeled?

Generally speaking, requirements in the automotive domain are documented in specification documents containing an informal textual description (cf. [Sikora et al. 2012]). This description is then supplemented by formal behavioral models such as MATLAB or ASCET for individual aspects. As described in Part II of this book, the requirements viewpoint (see Chapter 4) provides modeling concepts and language elements for:

- Static descriptions of requirements using context diagrams
- Dynamic descriptions of requirements using scenarios
- Traversal to the solution domain using function models

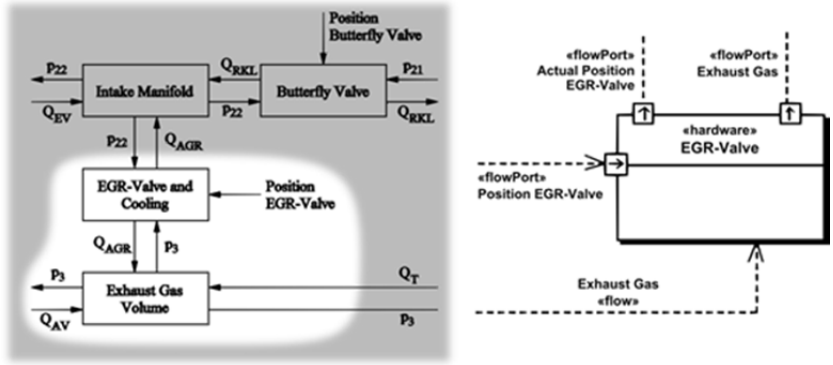


Fig. 12-1 *Comparing the plant structure (left) and the requirements viewpoint (right)*

The requirements viewpoint was evaluated in the context of the case study *Engine control system*. The aim of the evaluation was to identify which relevant domain characteristics are adequately supported and whether or not the efficiency and quality of development can be improved. For this purpose, context elements were compared to their modeling counterparts in the model of the requirements viewpoint and were evaluated with regard to the desired goals.

Does the approach cover relevant problem classes?

As a typical subsystem in an engine control system, the air system is characterized first and foremost by the system that is to be controlled. The context of the air system can be described from both a static perspective as well as a dynamic behavioral perspective using the plant, the user, the environment, and the control system. The individual context elements are characterized as follows:

- The context element *plant* represents the physics of the air system as a behavioral model in the form of differential equations. In order to specify requirements for the control of the plant, the target parameter we are striving to control (in this case the amount of air provided to the engine for combustion) must be put in relation over time along with other conditions (in this case, for example, outside air pressure). Additionally, the behavioral model that is synthesized from the physical equations of the system's components must be available. On closer inspection, the air system's component topology varies greatly between different operation phases, and thus the behavioral model varies as well. As a result, the operating air system presents itself as a multitude of variants that must respectively fulfill differing requirements in minute detail. The change between topologies occurs either due to external events or constellations within the system in

order to achieve required target values. In order to be able to control the target values, we need to know which information (sensor values) we have at our disposal and which physical parameters (actuators) we can influence. In our example, these are the throttle and the exhaust gas recirculation valve.

- The *usage* describes the intent with which the air system shall be operated and specifies the desired air amount that the air system control shall achieve. For example, the air system is operated using a specific pre-defined mix of fresh air and recirculated exhaust gas when regenerating the exhaust filter. Target value generation is continuous but may involve jumps when the operating situation changes spontaneously.
- The *environment* specifies information such as temperature, outside air pressure, battery voltage, and other environmental information relevant to the operation of the air system. These values are typically continuous.
- The *control system* involves describing information such as starting, operation, shutdown, or test modes as discrete events.

Evaluation of the static perspective

As described above, all value- and event-discrete elements can be adequately and completely described using a context diagram. Time- and value-continuous elements can be approximated by listing the main aspects of the continuous contextual element.

Evaluation of the dynamic perspective

Interactions between system and system context can be described very well using scenario diagrams. The question of which intents are used to operate the system is addressed particularly well. As far as continuous interactions between system and plant are concerned, this behavior must first be transferred to a discrete form. This only works well if the plant's behavior can be assumed to be continuous. While applicable for a wide range of operating situations of the air system, it cannot be assumed for all physical plants.

Addressing the problem classes in the context model

In summary, the question of an adequate problem description using the contextual view with the modeling concepts of the requirements viewpoint is depicted in Tab. 12-2. Value- and event-discrete behavior is addressed fully, whilst value- and time-continuous behavior can only be approximated through the use of simplifying discretization and thus remains incomplete.

Does the approach support the traversal from the problem domain to the solution domain?

Using value- and time-discrete semantics, behavior can be easily documented in an understandable form. For embedded systems, it is of utmost importance that they remain operational in all situations and thus expectations regarding completeness are extremely high. A scenario-based approach, however, is seldom complete or free from ambiguities. The value- and event-discrete semantics can only approximate value- and

time-continuous behavior and thus remain imprecise and, for our purposes, incomplete. The traversal to the solution domain relies on an approach based on value- and event-discrete sequences, ultimately leading to a state machine. If, however, as in our case, value- and time-continuous behavior is dominant, the solution must be based on a cybernetic control system approach utilizing signal-theoretic thinking.

Tab. 12-2 *Coverage of problem classes using the requirements viewpoint model*

Contextual Element	Problem Class	Requirements Viewpoint Model Type	Covered?
Plant	Value- and time-continuous	Context diagram	Partly
Environment	Value-discrete and -continuous	Context diagram Scenario diagram	Widely
Usage	Value-discrete Event-discrete	Context diagram Scenario diagram	Fully
System Control	Event-discrete	Scenario diagram	Fully

In order to be applicable to the domain of physically dominated systems, the requirements viewpoint model must be developed further in this direction. Under this premise, software can be developed based on system requirements resulting from systems co-design of all system disciplines concerned.

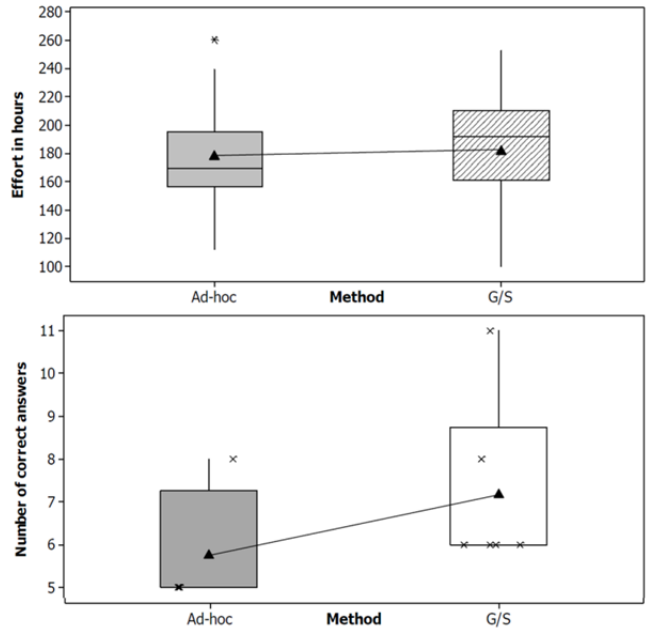


Fig. 12-2 *Effect on effort (top) and system comprehension (bottom)*

The following conclusions were reached by means of a case study [Gross et al. 2009] that was performed to assess the effect of the approach on efficiency and quality in the automotive domain (see Fig. 12-2). Firstly, the approach is neutral with regards to effort and quality (see Fig. 12-2, top). Secondly, the distribution of effort is shifted towards earlier development phases. Thirdly, communications between stakeholders and system understanding among developers were improved (see Fig. 12-2, right).

12.3.2 A Seamless Development Methodology for Automotive Systems

As seen in the evaluation of RQ1, the SPES engineering methodology has to be tailored for specific domains. In order to address RQ5, the SPES modeling framework was refined for the application within the automotive domain. We evaluated the concepts by means of the case study *Body control module*.

Requirements and functional viewpoints

Restricted natural language for requirements specification

As already stated in the last section, today, requirements are mostly specified in unrestricted natural language [Sikora et al. 2012]. The informal character of natural language, as well as its inherent ambiguity, can lead to inconsistent, ambiguous, and incomplete requirements. To resolve this problem, we use *requirement patterns* that are textual templates for different types of requirements [Kapeller and Krause 2006, Holtmann 2010]. Requirement types supported that reside in the requirements viewpoint (see Chapter 4) are, among other things, solution-oriented (see Section 4.2.4), timing, and safety requirements. However, functional requirement patterns describe the functional decomposition of the system under development (SUD) and reflect a functional hierarchy as explained in Chapter 5. Example 12-1 shows a functional requirement pattern in the upper part. The parts in square brackets are optional, and parts in angle brackets are variable and replaced by functionalities of the SUD. The lower part of Example 12-1 represents an instance of this requirement pattern and describes an excerpt of the functional decomposition of the SUD into its functionalities.

Example 12-1: Requirement pattern and instance

The functionality of the system "<system>" consists of the following function[s]: <function list>.

The functionality of the system "Control Indicators" consists of the following functions: Indicate, Switch Hazard Lights.

Processing of textual requirements for automated validation and transition to model-based design

Requirements formulated by means of requirement patterns are derived from the informal requirements systematically using a process refining the one used within the requirements viewpoint (see Section 4.4). Since we restricted the expressiveness of natural language and hence disambiguated it, the resulting requirements are understandable to all stakeholders and can be automatically processed at the same time. Thus, they can be validated automatically to ensure consistency or completeness, for example (see Fig. 12-3, left) [Holtmann et al. 2011a]. Furthermore, to ease the transition to model-based development, they are automatically transformed into a system analysis model using a Triple Graph Grammar (TGG) [Schürr 1995]. TGGs specify rules for bidirectional model-to-model transformations (M2M) that can be executed automatically and also preserve traceability and consistency. Requirements are formulated according to the requirement patterns and thus, the analysis model is kept traceable and consistent. By enabling the automatisms mentioned above, this procedure goes beyond the manual modeling of the SUD functions according to informal requirements as explained in Chapter 5.

Logical viewpoint

In the logical viewpoint, the logical component architecture is designed, as explained in Chapter 6, on the basis of the system analysis model resulting from the functional viewpoint. The elements of the analysis model are allocated to elements of the logical component architecture to document which logical components realize which functions and to maintain traceability [Meyer et al. 2011], see Fig. 12-3. For example, the functions *Indicate* and *Switch hazard lights* are allocated to two logical components, *Indicator* and *BrakeLampActuator*, respectively. The semantic correctness of the allocations can be ensured with the mapping relations based on contract-based design introduced in Part II.

Transition to AUTOSAR application software

To support a seamless development process, the logical component architecture can be automatically transformed into AUTOSAR application components. To do this, elements to be transformed into AUTOSAR application software (ASW) are marked manually using a

profile. The resulting refined logical component architecture is then automatically transformed using model-to-model transformations with TGGs (see Fig. 12-4) [Giese et al. 2010]. The usage of TGGs again allows for preservation of traceability and consistency between the models.

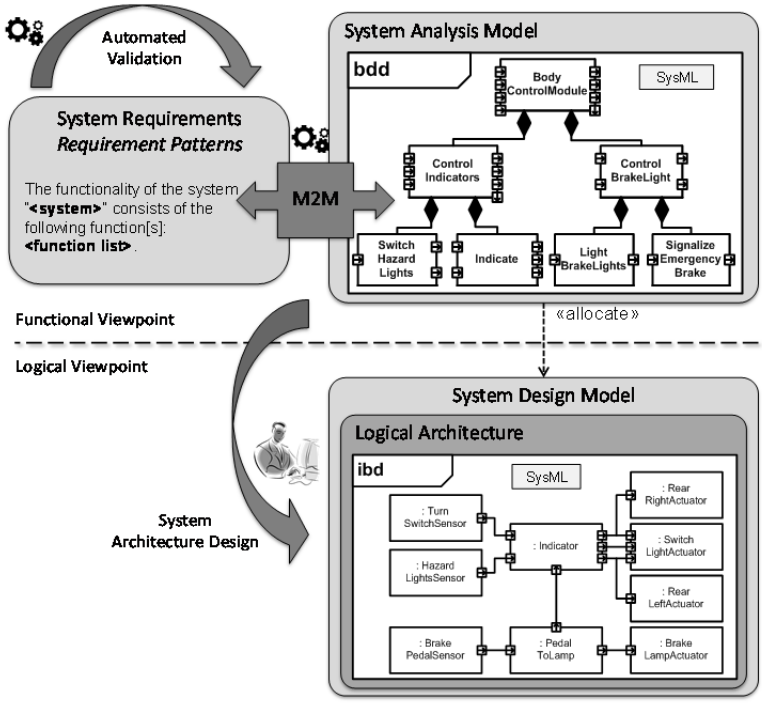


Fig. 12-3 *From the functional to the logical viewpoint*

Functional simulation of AUTOSAR application software

After the transformed AUTOSAR ASW has been manually enriched with behavior, this behavior can be simulated using the COTS tool SystemDesk[2] to validate its functionality before the software is deployed on hardware, see Fig. 12-4, upper right. Therefore, the compiled code of the overall SUD is executed, and thus its responses with regard to predefined stimuli are generated. The dynamic system behavior of the

[2] http://www.dspace.com/systemdesk/

interconnected software components can thus be tested at an early design stage, which addresses RQ6.

Technical viewpoint

Resource and real-time simulation

We further enrich the system design model with information from the technical viewpoint (see Chapter 7), as seen in Fig. 12-4, lower left. For example, a task calling operations of the logical component *Indicator*, its activation policy, and its allocation to an executing CPU are specified.

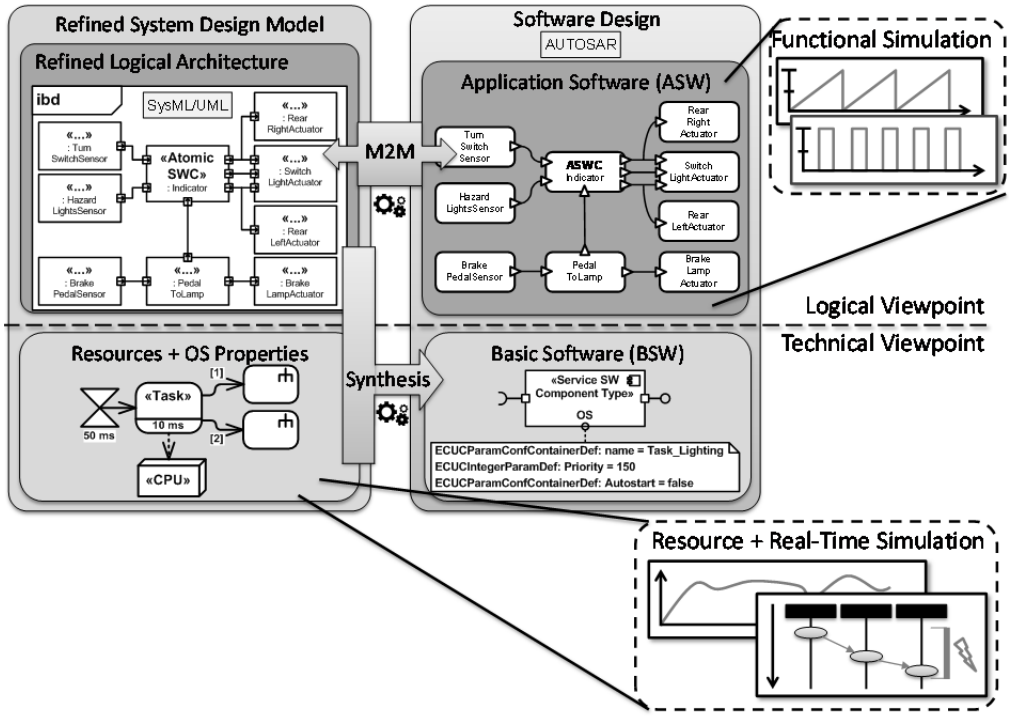


Fig. 12-4 *From the logical to the technical viewpoint*

On the one hand, we use this refined system design model to integrate a resource and real-time simulation of the SUD (see Fig. 12-4, lower right) to support architectural decisions and to validate the architecture

concerning timing requirements. This was realized by implementing a generator for simulation models for the COTS tool chronSIM[3] [Nickel et al. 2010], which allows computation of the CPU loads and simulation of the timing behavior of the SUD. Furthermore, timing requirements can be formalized by means of requirement patterns (see above) and used within the simulation to directly indicate requirement violations within the simulation [Meyer et al. 2011]. The simulation of the scheduling for a specific execution platform enables identification of design flaws in early development phases before first prototypes are available, addressing RQ6.

Partial synthesis of AUTOSAR basic software configurations

On the other hand, we use the refined system design model to further simplify the transition to AUTOSAR. In addition to the ASW, an AUTOSAR model consists of basic software (BSW) and an automatically generated middleware. Typically, for most parts the BSW is configured and then its source code is generated automatically. However, since there are thousands of different possible configurations depending on the architecture, support must be provided for this configuration task. In order to (semi-)automate the configuration, we developed an algorithm (see Fig. 12-4) for synthesizing parts of the AUTOSAR configuration [Meyer and Schäfer 2009]. For example, we preconfigure the operating system and the communication stack [Meyer and Holtmann 2011]. Thus, the main architecture decisions that were already specified within the refined system design model are transferred to the AUTOSAR basic software configurations.

Evaluation

In order to answer RQ8, the development methodology presented for automotive systems was evaluated with an excerpt of the case study *Body control module* by a proof of concept and structured expert interviews. For the evaluation, three experts from Hella and another industrial partner from the automotive sector were interviewed after using this new approach within a prototype. Of course, the number of persons interviewed is too small to gain a universally valid result, but we tried to minimize this effect by choosing experts for every development phase.

[3] http://www.inchron.com/chronsim.html

Furthermore, some concepts have already been applied in real development projects.

Consistency and traceability is maintained throughout the development process

The proof of concept demonstrated that the systematic and partially automated transitions between the different development phases and viewpoints, together with automated checks, ensure traceability and consistency throughout the methodology, especially if parts of an artifact in a development phase are added or removed. For example, newly added requirements result in new functions within the analysis model. In this case, the fact that there is no logical component that takes care of this function is revealed automatically. Furthermore, the expert interviews demonstrated that manual and thus extensive developer tasks are reduced by the transitions, and that the integration of simulation techniques in early development phases reduces extensive iteration loops.

12.3.3 Variant Handling

Embedded electronic systems are designed to fulfill not only the requirements of a single vehicle, but also the requirements of an entire family of vehicles. Therefore, as described in RQ9, the development methodology must address variant management in order to be applicable for the automotive domain. The information model for designing automotive embedded systems in the PREEvision tool is aligned to the SPES modeling framework and supports the systems engineering principles of abstraction, modularization, and reuse.

Information model for designing automotive embedded systems

Every SPES viewpoint can be assigned to one or several PREEvision modeling layers. In PREEvision, product lines of electronic vehicle systems are modeled from requirements to the hardware geometry using seven main layers.

SPES viewpoints and PREEvision modeling layers

On the top level, the design starts with the definition of requirements and customer features. These layers correspond to the SPES requirements viewpoint. Vehicle features can be defined in PREEvision using requirements. Customer features describe a set of features of the vehicle from the vehicle user's perspective. The customer features are organized in a tree representing a superset of features to be fulfilled by a vehicle product line. This is often referred to as a "150% model."

Example 12-2: Engine variants

All engine variants are considered in the product line, but only one engine variant will be delivered and finally built in a vehicle. All transmission variants are considered, but only one transmission will be selected later on. All optional features are considered, but only a subset will be chosen later on.

In practice, a two-step-approach for variant management is often applied — from a vehicle product line ("150% model") to a vehicle family (often also called "120% model") to a concrete vehicle ("100% model"). Due to the high number of options, it is not possible to manage each and every variant explicitly. Therefore, typically 150% models and 120% models are designed.

PREEvision supports the definition of variant conditions between features. Such variant conditions can be, for example, "exclusive-or" relations or "needs" relations between features, but also "or" and "optional" relations.

The result is a customer feature model (cf. Fig. 12-5) similar to the FODA approach [Kang et al. 1990]. In production projects, the customer feature tree is typically organized according to the vehicle manufacturer's organizational units following the classical division powertrain, chassis, body, comfort, and multimedia.

Fig. 12-5 *Customer feature model in PREEvision*

The next design level is the functional viewpoint, referred to as the logical architecture in PREEvision. Hierarchies are used to structure the complete layer according to the organizational responsibilities at the

vehicle manufacturer. However, logical signal connections crossing the organizational responsibilities can also be modeled by system diagrams — or activity chains can be used to express a logical control sequence from the sensors to the actuators realizing a given customer feature. This means that customer features are represented by an orthogonal subset of the organizational structure of the logical architecture.

The "realization" or "implementation" relations between customer features and logical architecture are specified using mappings. Mappings are information objects that decouple all the modeling layers. This enables the design engineer to handle even variants of mappings.

Example 12-3: Mapping variants between logical architecture and the hardware architecture

In practice, the logical architecture and the hardware architecture are often mapped to each other in a variant-specific manner. In this case, only the mappings are variant-specific — the logical and the hardware architecture are neutral.

All subsequent modeling layers in PREEvision are used to specify the logical and the technical viewpoints in the SPES modeling framework. Here, PREEvision differentiates between system software architecture to model the logical viewpoint and the software implementation and hardware architecture to model the technical viewpoint.

The system software architecture supports the AUTOSAR platform [AUTOSAR 2011], including the implementation of software components. The hardware architecture is modeled in several abstractions representing the hardware network topology, the hardware components, the schematics, and wiring harness. Geometry data are stored in the technical viewpoint.

The communication layer supports the definition of conventional and bus signals, protocol data units, frames, and communication schedules.

Variant management

A product variant is defined in PREEvision as a consistent subset of a product line over all layers. Challenges from a tool perspective are the consistency and completeness of the defined subsets: they have to be guaranteed inside every layer, but also across the layers.

Questions such as "Are all input signals needed provided in a given variant?" or "Are all software components that are part of a given variant mapped to hardware components?" have to be answered with "yes." This can be checked by user-defined consistency checks and propagation rules

that can be designed on a customer-specific basis — and that guarantee consistent and complete variant models.

The approach also has to be usable by an organization following a defined process with distributed responsibilities and defined roles. Examples for roles are persons responsible for software, persons responsible for hardware, and system architects. Database functions are available in PREEvision to prevent concurrent access to an artifact, but also conveniently support concurrent work on different artifacts, history, and archive functions.

Variant design is supported by engineering features such as the signal router taking into account only a selected variant and not always the complete product line. Highlighting of variants is supported in all diagrams, and metrics are available to calculate characteristics of a product variant such as bus loads, weight, or costs.

We succeeded in developing concepts for the integration of Simulink as an implementation tool for software components. Furthermore, usability, convenience, and adaptation functions were designed for the usage of PREEvision in production projects. Extended feature modeling capabilities were developed and the support of distributed responsibilities and database functions was optimized. Further research topics included consistency analysis not only inside a given layer, but also across layers.

12.4 Summary

Several challenges arise due to characteristics inherent to the automotive domain. Although all aspects of development are addressed by processes employed, significant gaps are often present between process steps. This also applies to the artifacts these steps respectively produce or use, further affecting traceability between them. Requirements are usually documented in informal natural language, introducing the possibility of ambiguity, inconsistency, and incompleteness, and must be considered by formal models used in later design phases. Embedded systems must cope with continuous plant behavior that is both difficult to specify as well as validate, and even more difficult to control, using systems tailored toward discrete behavior. Furthermore, embedded systems in the automotive domain are becoming more complex and it is therefore difficult to validate in early design phases whether the specified architecture is correct. Additionally, systems in the automotive domain are usually designed for entire product lines, making variant management important for handling the resulting complexity of development.

Many of these aspects have now been addressed and at least partly solved by means of the methodologies developed to answer the research questions presented in Section 12.2. As a whole, the SPES methodology is a very good basis, but also leaves room for further development for refining and adapting the approaches for specific application domains.

12.5 References

[AutomotiveSIG 2010] Automotive Special Interest Group (SIG): Automotive SPICE. Process Reference Model. Accessed on: April 3, 2012. http://www.automotivespice.com/automotiveSIG_PRM_v45.pdf.

[AUTOSAR 2011] AUTOSAR GbR: Specification of ECU Configuration. http://www.autosar.org/download/AUTOSAR_ECU_Configuration.pdf. Accessed on April 3, 2012.

[Giese et al. 2010] H. Giese, S. Hildebrandt, S. Neumann: Model synchronization at work: Keeping SysML and AUTOSAR models consistent. In: G. Engels, C. Lewerentz, W. Schäfer, A. Schürr, B. Westfechtel (Eds.): Graph Transformations and Model-Driven Engineering. Springer, Berlin/Heidelberg, 2010; pp. 555–579.

[Gross et al. 2009] A. Gross, J. Dörr, I. Menzel, M. Müller: Use Cases vs. Funktionale Spezifikation: Ein experimenteller Vergleich zweier Techniken zur Anforderungsspezifikation, GI-Fachgruppen-Treffen Requirements Engineering, 2009.

[Holtmann 2010] J. Holtmann: Mit Satzmustern von textuellen Anforderungen zu Modellen. In: OBJEKTspektrum, Vol. RE/2010, 2010, http://www.sigs-datacom.de/fileadmin/user_upload/zeitschriften/os/2010/RE/holtmann_OS_RE_2010.pdf. Accessed on: March 31, 2012.

[Holtmann et al. 2011a] J. Holtmann, J. Meyer, M. von Detten: Automatic validation and correction of formalized, textual requirements. In: Proceedings of the IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops (ICSTW) 2011. IEEE Computer Society, Los Alamitos, 2011, pp. 486–495.

[Holtmann et al. 2011b] J. Holtmann, J. Meyer, M. Meyer: A seamless model-based development process for automotive systems. In: R. Reussner, A. Pretschner, S. Jähnichen (Eds.): Software Engineering 2011 – Workshopband (inkl. Doktorandensymposium). Bonner Köllen Verlag, Bonn, 2011, pp. 79–88.

[Kang et al. 1990] DTIC: Feature-Oriented Domain Analysis (FODA) Feasibility Study. http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA235785. Accessed on April 3, 2012.

[Kapeller and Krause 2006] R. Kapeller, S. Krause: So natürlich wie Sprechen – Embedded Systeme modellieren. In: Design & Elektronik, 2006/08; pp. 64–67.

[Meyer and Holtmann 2011] J. Meyer, J. Holtmann: Eine durchgängige Entwicklungsmethode von der Systemarchitektur bis zur Softwarearchitektur mit AUTOSAR. In: H. Giese, M. Huhn, J. Philipps, B. Schätz (Eds.): Tagungsband des Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme VII. fortiss, Munich, 2011, pp. 21–30.

[Meyer and Schäfer 2009] J. Meyer, W. Schäfer: Automatische Analyse und Generierung von AUTOSAR-Konfigurationsdaten. In: H. Giese, M. Huhn, U. Nickel, Bernhard

Schätz (Eds.): Tagungsband des Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme V. Carl-Friedrich-Gauß-Fakultät für Mathematik und Informatik, Technische Universität Braunschweig, 2009, pp. 82–91.

[Meyer et al. 2011] J. Meyer, J. Holtmann, M. Meyer: Formalisierung von Anforderungen und Betriebssystemeigenschaften zur frühzeitigen Simulation von eingebetteten, automobilen Systemen. In: J. Gausemeier, F. Rammig, W. Schäfer, A. Trächtler (Eds.): 8. Paderborner Workshop Entwurf mechatronischer Systeme. Heinz Nixdorf Institut, Paderborn, 2011, pp. 203–215.

[Nickel et al. 2010] U. Nickel, J. Meyer; T. Kramer: Wie hoch ist die Performance?. In: Automobil-Elektronik, Vol. 2010, No. 3, 2010, pp. 36–38.

[Schürr 1995] A. Schürr: Specification of graph translators with triple graph grammars. In: E. W. Mayr (Eds.): Graph-Theoretic Concepts in Computer Science. Lecture Notes in Computer Science, Vol. 903, Springer, Berlin/Heidelberg, 1995, pp. 151–163.

[Sikora et al. 2012] E. Sikora, B. Tenbergen, K. Pohl. Industry needs and research directions in requirements engineering for embedded systems. In: Requirements Engineering Journal, Vol. 17, No.1, 2012, pp. 57-78.

[Zimmer et al. 2011] B. Zimmer, S. Bürklen, M. Knoop, J. Höfflinger, M. Trapp: Vertical safety interfaces - improving the efficiency of modular certification. In: Proceedings of the 30th International Conference of Computer Safety, Reliability, and Security, 2011.