Marian Daun
Bastian Tenbergen
Dr. Thorsten Weyer

4

Requirements Viewpoint

The requirements viewpoint defines concepts and techniques for systematically eliciting and specifying the requirements for a system under development. The requirements viewpoint differentiates between different artifact types that document different information elicited during requirements engineering:

- *Context, which documents the operational environment in which the system under development is embedded*
- *Goals, which document stakeholder intentions with regard to the system under development*
- *Scenarios, which document typical interactions between the system under development and its context*
- *Solution-oriented requirements, which document the requirements for the system under development in a precise and complete manner*

K. Pohl et al. (eds.), *Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology*,
DOI 10.1007/978-3-642-34614-9_4,

4.1 Introduction to the Requirements Viewpoint

Separation of requirements and solution

The requirements viewpoint comprises the part of the SPES modeling framework that primarily deals with the accurate, complete, and consistent specification of system requirements. These requirements serve as input for functional analyses and architecture design (see Chapters 5 to 7). The goal of the requirements viewpoint is to:

- Gain a comprehensive understanding of the system under development
- Foster the best possible freedom in development by preventing premature commitment to possible solutions
- Supply the necessary information such that decisions pertaining to concrete implementation can be made during subsequent architecture design

Solution-neutral and solution-oriented requirements; co-design process

In the requirements viewpoint, a strict separation between stakeholder intentions and solution-oriented requirements is maintained. In order to support this separation, the requirements viewpoint contains three solution concepts:

- *Solution-neutral requirements* describe the intentions of the stakeholders and the added benefit that can be gained for the stakeholders [Leveson 2000]. Concrete aspects of a possible solution are ignored.
- *Solution-oriented requirements* describe necessary properties of operations, system states, and the information structure, as well as qualities that a solution must possess [Pohl 2010]. Solution-oriented requirements are the connection between solution-neutral requirements and concrete implementations.
- The *intertwined development* of requirements artifacts is based on a goal-/scenario-oriented, step-by-step refinement of requirements from solution-neutral to solution-oriented requirements. Due to the step-by-step, artifact-based refinement, the intertwined development allows for traceability between requirements artifacts, ensures requirements consistency between the artifacts, and leads to completeness with regard to the requirements artifacts and the requirements specification.

Artifact model, types of requirements models

The requirements viewpoint documents a complete system requirements specification by means of partial diagrams. Therefore, each artifact model contains a number of different requirements diagrams (see Section

4.2). Due to the number of different requirements artifacts and their interrelations, the requirements viewpoint is very comprehensive as well as complex. Therefore, different model types are used. By using different model types within the requirements viewpoint, the corresponding view is constructed by integrating each model type based on common facts. Typical model types are goals and scenarios, as well as structural, operational, and behavioral models (see [Pohl 2010] and Section 4.2.4).

The artifact model of the requirements viewpoint is explained below (Section 4.2). In addition, the integration of the requirements viewpoint with other viewpoints and abstraction layers of the SPES modeling framework is illustrated in Section 4.3. Finally, we outline a requirements engineering process across several abstraction layers: it can be used to systematically develop requirements of the system under development (SUD) and can be tailored for individual project needs (Section 4.4).

4.2 Requirements Artifacts

In this section, we briefly outline the artifact model of the requirements viewpoint. Each subsection outlines one artifact type and gives a short example.

4.2.1 Context Model

Importance of the system context

The context of the system is that part of the operational infrastructure that does not belong to the system (and therefore cannot be influenced during development) but surrounds the system once it has been deployed (and therefore strongly impacts the definition of requirements for the SUD). If the context of the SUD is not properly understood, it is impossible to properly define and interpret the requirements for the SUD [McMenamin and Palmer 1984, Davis 1993, Jarke and Pohl 1994, Hammond et al. 2001]. The requirements viewpoint therefore contains context models for modeling that part of the environment that influences the system. Context models can be used to document constraints from the physical environment of the system that limit the scope, solution space, or development process (e.g., the environment it will be deployed in, company-specific regulations, or laws and legislation that must be adhered to).

Context entities, interfaces, and system interaction

Context models focus on the system's desired interaction with its environment or, more precisely, its context entities [Weyer 2011]. Context entities are, for example, external actors, sensors, and other

systems in the environment that interact with the SUD. Each specified context entity must be present in at least one scenario model (see Section 4.2.3) so that the SUD's interaction with each entity can be assessed. Context models allow system goals to be determined and give a first impression about the SUD's interaction with its context. In the SPES modeling framework, context models also define the interfaces of the functional black box model in the functional viewpoint (see Section 5). Further information on context models can be found in [Weyer 2011]. In the requirements viewpoint of the SPES modeling framework, a number of different types of context models can be used. For example, structural diagrams such as SysML block definition diagrams or internal block diagrams [OMG 2010a] can be used to document static/structural context information. On the other hand, dynamic aspects of the context can be documented using Petri nets [Reisig 1991] or communicating finite state machines [Lynch and Tuttle 1989, Alfaro and Henziger 2001].

Example of a context model

Fig. 4-1 shows an example of the context model of a simplified automation system as a SysML block definition diagram. The SUD (the <<block>> stereotype in the middle) is treated as a black box, that is, no internal properties are considered. There are a number of context entities (<<actor>> stereotypes) that communicate with the SUD, either receiving output from the SUD or producing input to the SUD.

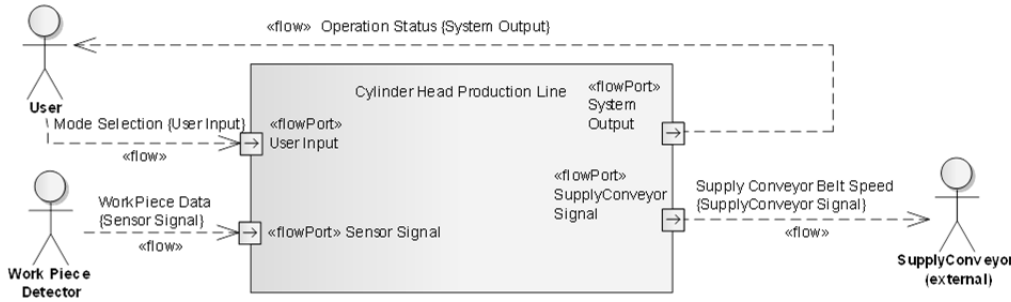

Fig. 4-1 *Example of a context diagram* [1]

[1] All figures in the requirements viewpoint have been modeled using Enterprise Architect®

Hint 4-1 lists the rules that have been defined in the requirements view for ensuring that the context of the SUD has been modeled completely and correctly.

Rules for checking context models

Hint 4-1*: Rules for checking context models*

- ❑ Have all actors in the system context that receive output from or produce input to the SUD been considered?
- ❑ Have all inputs that the SUD receives from the environment or from entities within the system context been considered?
- ❑ Have all outputs that the SUD delivers to the system context or to context entities been considered?

4.2.2 Goal Model

Capturing stakeholder intentions

Goal models document the intentions of stakeholders when they are conceiving the system. They represent a first manifestation of the stakeholders' system vision. Goals give rationales and justifications for the functionalities and features the system must possess. Goals ignore concrete aspects of the solution and hence serve as an essential means for negotiating requirements and their necessity with regard to the system envisioned. The purpose of negotiating requirements on the basis of goals is to establish a common understanding of the envisioned among all stakeholders. In addition, goals can be used to document necessary quality aspects such as the system's safety features (see Chapter 8) or real-time behavior (see Chapter 9) that in turn will be specified using solution-oriented requirements (see Section 4.2.4).

Goal types and goal refinement

In goal models, relationships can be identified between goals, functions, and qualities. For example, goals might be in direct conflict with one another (i.e., fulfilling one goal will make it impossible to fulfill a conflicting goal), or the fulfillment of goals may contribute positively or negatively to the fulfillment of another goal (i.e., make it easier or harder to achieve the other goal). In addition, goals can be refined using AND and OR refinements: AND refinements denote that a number of refining goals have to be fulfilled in order to fulfill the refined goal; OR refinements denote that at least one of the refining goals has to be fulfilled in order to fulfill the refined goal. Furthermore, we can distinguish between hard and soft goals. Hard goals are goals whose fulfillment can be verified by means of simple yes/no checks (i.e., either the goal has been fulfilled or not). In contrast, soft goals represent goals that the system to be developed fulfills to a certain degree.

Sources of goals/stakeholder intentions

Goals can be determined in part from the context model but also through stakeholder collaboration. Typical stakeholders who may contribute goals to a system are clients, contractors, product managers, business managers, technical leaders, certifiers or certifying authorities, or the legislative authority. Goals and goal modeling are explained in more detail in [Yu 1997, Lamsweerde 2009, Pohl 2010]. In the requirements view of the SPES methodology, KAOS goal diagrams, i* models, or SysML requirements diagrams can be used to model this artifact type.

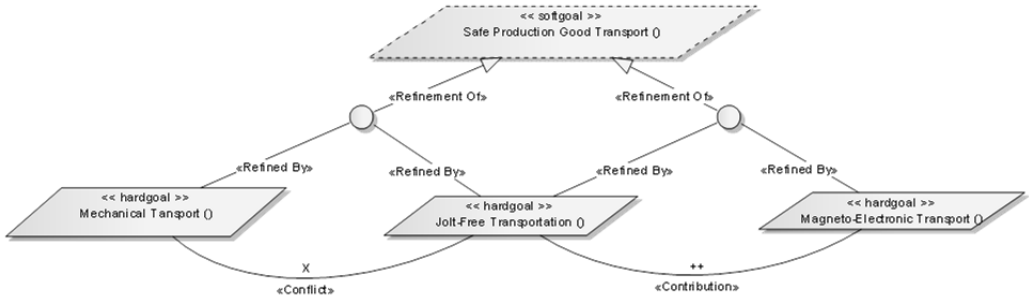

Fig. 4-2 *Example of a goal diagram*

Example of a goal diagram

Fig. 4-2 shows the goal diagram of an example system using the KAOS notation [Lamsweerde 2009]. The goals are structured hierarchically through AND and OR refinement. In this diagram, the top-most soft goal is refined by means of two alternatives (OR refinement). One alternative consists of two hard goals that both have to be fulfilled for the soft goal to be fulfilled (AND refinement). However, there is a conflict between these two goals, which may indicate that this alternative is not a suitable refinement of the soft goal. The other alternative also consists of two hard goals (one of which is also a refinement of the other alternative) that both have to be fulfilled for this alternative to be a valid refinement of the soft goal (AND refinement). The diagram shows a contribution link between the two goals in this alternative, indicating that the fulfillment of one goal positively contributes to the fulfillment of the other goal. Hence, this alternative is preferable over the other alternative.

Hint 4-2 lists the rules that have been defined in the requirements viewpoint for ensuring that all goals for the SUD have been modeled completely and correctly.

Hint 4-2: *Rules for checking goal models*

- ❑ Have all hard goals of stakeholders been captured?
- ❑ Have all soft goals of stakeholders been captured?
- ❑ Have all abstract (hard or soft) goals been refined using AND and OR refinements?
- ❑ Have all positive and negative contributions from one goal to another goal been uncovered and documented?

Rules for checking goal models

4.2.3 Scenario Models

Example interactions with the system

Scenarios specify example interactions of the system with its context. They allow requirements to be determined by modeling the system's interaction with context entities that have been identified in the context models (see Section 4.2.1). This enables the system's benefit and impact on the system context to be assessed. The actors that are present in any scenario model must be present in at least one context model that has been specified earlier. Scenarios fulfill the goals that have been specified in the goal models (see Section 4.2.2). In the requirements viewpoint, any goal has to be fulfilled by at least one scenario and every scenario must fulfill at least one goal. Scenario execution is typically constrained by preconditions. After scenario execution, specific postconditions must hold for the entire system. Furthermore, scenarios may specify some internal states that can be used to draft an initial specification of the behavioral requirements models of solution-oriented requirements (see Section 4.2.4). In scenario models, similarly to the goal models, the system is considered as a black box. Hence, there must not be any indication within either model that depicts the internal structure of the SUD. We can distinguish between different types of scenarios, for example:

- ❑ *Main scenarios:* Main scenarios describe the standard way of fulfilling one or more goals.
- ❑ *Alternative scenarios:* Alternative scenarios describe alternative ways of fulfilling the same goals as in the corresponding main scenario. Alternative scenarios may also be used for error handling in cases in which the associated goals can still be fulfilled.
- ❑ *Exception scenarios:* Exception scenarios describe how the system must react in the case of a critical error during scenario execution that prevents fulfillment of the associated goal. Exception scenarios place particular emphasis on error recovery rather than on goal fulfillment.

Structuring scenarios

Additional information on scenario modeling can be found in [Pohl 2010] and [Potts 1995]. In the requirements viewpoint of the SPES modeling framework, SysML sequence diagrams [OMG 2010a] or ITU message sequence charts [ITU 2004] can be used to model this artifact type. During the requirements engineering process, it may be useful to model multiple scenarios. Scenarios can be structured using use cases ([OMG 2010a, OMG 2010bCockburn 2001], and use cases can be related to one another, for example, by means of include and extend relationships) or hMSCs [ITU 2004]. However, when using structuring scenarios in this way, the scenario specification must therefore document a complete behavioral specification.

Fig. 4-3 shows a SysML sequence diagram with a scenario model. The diagram depicts a scenario for executing a production process. This scenario fulfills one goal from Fig. 4-2. Furthermore, the model in Fig. 4-3 specifies five states that the SUD adopts during this interaction (for details, see Section 4.2.4).

Hint 4-3 lists the rules that have been defined in the requirements viewpoint for ensuring that the scenario artifacts have been modeled completely and correctly.

Rules for checking scenario models

Hint 4-3*: Rules for checking scenario models*

- ❑ Has a precondition been specified for each scenario?
- ❑ Does every scenario describe the entire interaction necessary to fulfill one or more goals?
- ❑ Does every scenario account for all actors that interact with the system?
- ❑ Have postconditions been specified for every scenario?

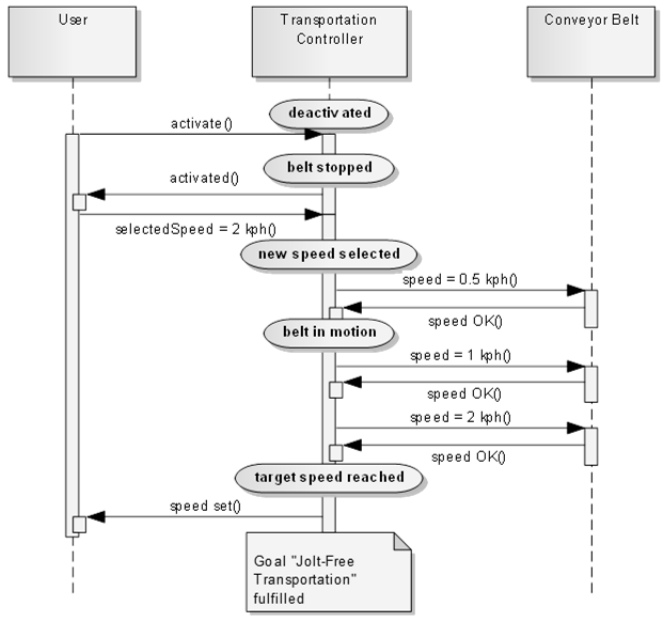

Fig. 4-3 *Example of a sequence diagram*

4.2.4 Solution-Oriented Requirements Model

Solution-oriented requirements are solution-specific descriptions of behavior, operations, and the information structure of the solution concept developed (see [Pohl 2010] and [Davis 1993]). They thus represent a first step towards the implementation. Solution-oriented requirements consist of a structural requirements model, an operational requirements model, and a behavioral requirements model. Solution-oriented requirements can thus be derived from scenario descriptions as scenarios may specify states that the SUD adopts after a certain interaction sequence has been executed. Furthermore, the operational requirements model and the structural requirements model of solution-oriented requirements can be derived in part based on scenarios and the context model, as both specify information that is exchanged between the SUD and the context and show how information is transformed from input to output.

Complementary development of the three perspectives

All three types of solution-oriented requirements models are developed complementarily as they present separate but interrelated aspects of the same SUD. A more detailed explanation of solution-oriented requirements is given in [Pohl 2010].

In the requirements viewpoint of the SPES modeling framework, SysML block definition diagrams can be used as static/structural models,

SysML activity diagrams can be used to model operational requirements models, and SysML state machine diagrams can be used to model behavioral requirements models. In the following sections, an example is given for each model type along with a brief explanation and the rules for checking each model type.

Structural Requirements Model

Purpose and example of structural requirements models

Fig. 4-4 shows an information structure model for an example system as a SysML block definition diagram. As shown, the static/structural requirements model gives a closer account of the information that is exchanged along the interfaces in the context model (see Section 4.2.1) and in part by the scenario model (see Section 4.2.3). Static/structural requirements models must therefore be defined consistently to both artifacts and can be used to document relationships between the objects pertaining to the information structure and other artifacts. For example, if a context model specifies the object "work piece data" to be exchanged between the SUD and its context, structural requirements models can be used to refine what information item "work piece data" consists of, e.g.: material type, length, width, height, and weight.

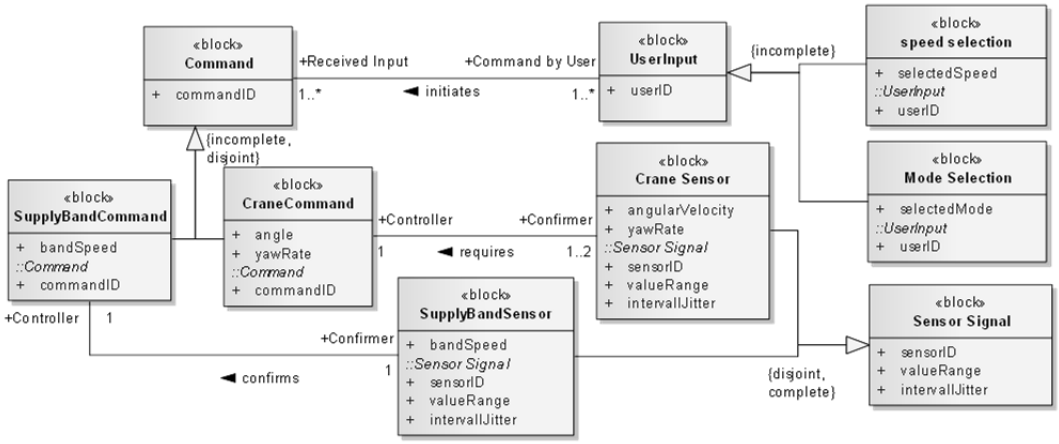

Fig. 4-4 *Example of an information structure diagram*

Hint 4-4 lists the rules that have been defined in the requirements viewpoint for ensuring that the structural requirements models have been modeled completely and correctly.

Hint 4-4: *Rules for checking structural requirements models*

- ❑ Have all inputs to the system from the context and its context entities (as specified in the context and scenario models) been accounted for?
- ❑ Have all outputs from the system to the system context and context entities (as specified in the context and scenario models) been accounted for?
- ❑ Have all information structures that are specified in behavioral and operational requirements models been documented?
- ❑ Have useful, non-trivial relationships (such as generalizations, aggregations, compositions) been introduced between information objects?

Rules for checking structural requirements models

Operational Requirements Model

Purpose and example of operational requirements models

Fig. 4-5 shows a SysML activity diagram as an example of an operational requirements model. This artifact type models operations that are derived by assigning user functions to the goals specified in the goal models (see Section 4.2.2) with reference to the interactions specified in the scenario models (see Section 4.2.3). Operational requirements models can therefore be seen as the solution-specific counterpart of the solution-neutral scenario artifacts. Consequently, the operations specified in the operational requirements models implement the functionalities that can be experienced by context entities (i.e., actors or external systems) through the interfaces that the system has with the context entities. As a result, the interfaces specified herein must be consistent to the interfaces specified in context models (see Section 4.2.1). This is similar to the functional black box model in the functional viewpoint (see Section 5), however, in contrast to the functional viewpoint, operational requirements models are partial requirements models that document the system's interaction with the context in more detail than scenario models. On the other hand, the functional viewpoint documents the entirety of the system's functions in order to foster analysis. As a consequence, artifacts specified in the functional viewpoint are based on the solution-oriented requirements models, particularly on the operational requirements models.

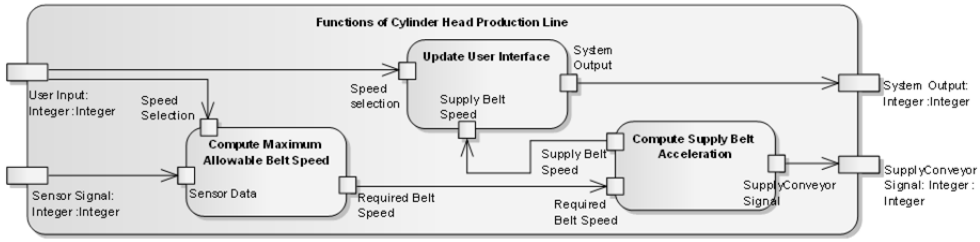

Fig. 4-5 *Example of an operational requirements diagram*

Hint 4-5 lists the rules that have been defined in the requirements viewpoint for ensuring that the operational requirements models have been modeled completely and correctly.

Rules for checking operational requirements models

Hint 4-5*: Rules for checking operational requirements models*

- ❑ Have all relevant system functionalities that have to be implemented by the SUD to fulfill its goals been considered?
- ❑ Have inputs and outputs been defined for every operation in the operational requirements models?
- ❑ Are the specified interfaces consistent to the interfaces in the context models?

Behavioral Requirements Model

Purpose and example of behavioral requirements models

Behavioral requirements models can be used to specify preconditions that must be in effect for system operations to be executed or postconditions that have to be fulfilled after an operation has been executed. Fig. 4-6 shows an example of a behavioral requirements model as a SysML state machine diagram. In this diagram, the states were partially derived from the scenario models (see Section 4.2.3). Transitions were also derived from the scenario models and completed during the specification of the requirements artifact at hand. Since the white box model of the functional viewpoint (see Chapter 5) uses state machines for specifying the internal behavior of functions, those behavioral models are based on the behavioral requirements models of the requirements viewpoint.

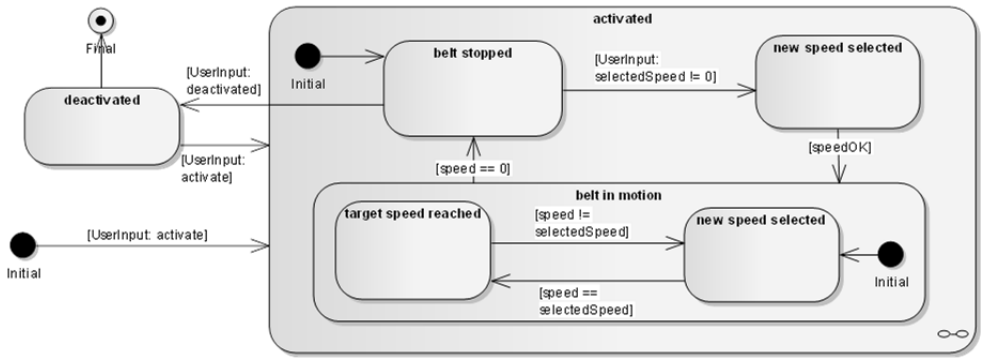

Fig. 4-6 *Example of a behavioral requirements diagram*

Hint 4-6 lists the rules that have been defined in the requirements viewpoint for ensuring that the behavioral requirements models have been modeled completely and correctly.

Rules for checking behavioral requirements models

Hint 4-6: *Rules for checking behavioral requirements models*

- ❑ Have all trigger events been considered in the behavioral requirements models?
- ❑ Have all system states and transitions of the SUD been considered?
- ❑ Do the behavioral requirements models specify preconditions and postconditions for scenarios?
- ❑ Do the behavioral requirements models specify activation conditions for operations?

4.3 Integration in the SPES Modeling Framework

This section gives an account of why some model types can be used in multiple viewpoints and how these model types have to be interpreted in the viewpoints (Section 4.3.1). Furthermore, this section explains how the requirements viewpoint can be integrated into the SPES modeling framework with regard to other viewpoints (Section 4.3.2) and different abstraction layers (Section 4.3.3). Hint 4-7 gives a short summary of correspondence rules that ensure consistency between the artifacts developed in each step.

Correspondence rules for the models in the requirements viewpoint

Hint 4-7*: Correspondence rules*

Context models ↔ scenario models

- ❑ Each actor included in any scenario has to be included in the context model
- ❑ Each actor included in the context model has to be included in at least one scenario model
- ❑ Inputs and outputs between the SUD and any actor have to be consistent in context and scenario models

Goal models ↔ scenario models

- ❑ Each scenario has to be related to at least one goal
- ❑ Each goal has to be fulfilled by at least one scenario

Goal models ↔ requirements models

- ❑ For each goal, system properties (functions, behavior, information structures) have to be defined in the requirements models related to the goal for fulfillment
- ❑ Each property documented in the requirements models must be related to at least one goal

Scenario models ↔ requirements models

- ❑ Each scenario must be capable of being processed based on the requirements models
- ❑ The inputs and outputs from the scenario models have to be consistent with the requirements models

Between requirements models

- ❑ The entry condition for each function defined in the function model has to be defined in the behavior model
- ❑ The information structure of the inputs and outputs of functions in each function model have to be defined in the information structure model
- ❑ The state-based actions and the transition-based actions belonging to the behavior model have to be described as functions in the function model

4.3.1 Use of Models across Viewpoints

Same model type, different viewpoints

The various model types used in the requirements viewpoint are also used in the other viewpoints. However, depending on the viewpoint, the model types have vastly different meanings and document entirely different information. For example, if a statechart were used in both the requirements viewpoint and the logical viewpoint, the statechart in the requirements viewpoint would represent the captured requirements and would summarize a possible solution with regard to the requirements. On the other hand, in the logical viewpoint, the statechart would represent a part of the logical architecture and it would detail how the system will be implemented rather than how it could be implemented. Similarly, functional models are used both in the functional viewpoint and in the requirements viewpoint. While in the requirements viewpoint operational

requirements models represent a type of partial functional model, the functional viewpoint is more concerned with an integrated, functional view on the entire SUD.

4.3.2 Integration across Viewpoints

Requirements viewpoint is the starting point for development

The requirements viewpoint is the starting point for the development process using the SPES modeling framework. Once requirements engineering activities in the requirements viewpoint have reached a satisfactory stability, the development process continues with the activities in the functional viewpoint (see Chapter 5). The functional viewpoint takes the scenario models and the solution-oriented artifacts from behavioral and functional requirements models as input and derives an approximate functional architecture that meets the requirements outlined in the artifacts. Further input from the requirements viewpoint is given to the logical and technical viewpoints (see Chapters 6 and 7 respectively). The logical viewpoint takes the system context and goal artifacts from the requirements viewpoint and derives a logical architecture that meets quality requirements defined in these requirements viewpoint artifacts. These artifacts also provide quality requirements for the technical viewpoint. In addition, the technical viewpoint suggests a concrete hardware/software architecture based on the requirements, functional, and logical viewpoints.

4.3.3 Integration across Abstraction Layers

All requirements artifacts on all abstraction layers

One key feature of the SPES modeling framework is the hierarchy of abstraction layers (see Section 3.4.1). Specifying requirements on different abstraction layers is a proven approach to reducing the complexity of development projects [Braun et al. 2010]. The requirements viewpoint therefore allows specification of all requirements artifacts. At each abstraction layer, the same set of artifacts is developed (i.e., context models, goal models, scenario models, and solution-oriented requirements models; see Section 4.2). The abstraction layers differ from one another with regard to the level of detail contained within their respective requirements artifacts, such that some abstraction layers contain more coarsely specified requirements (in the following, called higher abstraction layers) and some layers contain more detailed requirements (lower abstraction layers).

SUD decomposition in other viewpoints

The logical and/or technical viewpoints structurally decompose the SUD into subsystems. The decomposed subsystems that are structurally significant (e.g., important control units or safety-critical subsystems)

become the new focus of development and are hence treated as if they were the SUD on the next lower abstraction layer. The requirements process (see Section 4.4) starts anew for all of these subsystems.

4.4 The Requirements Process Model across Abstraction Layers

The following briefly illustrates an idealized development process that outlines the development of the different artifacts over time.

Step 1: Analyze and document the system context

- *1st Step: Analyze and document the system context:* Firstly, the system context in which the SUD will be used is analyzed and documented. The context of any subsystem consists of relevant parts of the context of the SUD as well as other subsystems of the SUD that the subsystem under development interacts with.

Step 2: Analyze and document goals

- *2nd Step: Analyze and document goals:* After modeling the system context, goals for the subsystem under development are elicited, documented, and negotiated with the stakeholders identified during context analysis. For the development of subsystems specifically, the documented goals must be consistent with those documented for the SUD. In detail, this means that the fulfillment of the goals of the SUD is dependent on the fulfillment of all goals of all of its subsystems.

Step 3: Define and model scenarios

- *3rd Step: Define and model the scenarios of system usage:* After the context and goal models have been sufficiently documented, scenarios are used to describe possible ways to fulfill the goals. The scenarios and goals have to be related: each goal has to be fulfilled by at least one scenario and each scenario must fulfill at least one goal. The development of goals and scenarios is a highly iterative and incremental process. Scenarios may lead to further goals not discovered in the first step. New goals will lead to further scenarios. This process continues until no new goals or scenarios are discovered. Scenarios of any subsystems depict refinements of scenarios of the SUD.

Step 4: Specify solution-oriented requirements

- *4th Step: Specify solution-oriented requirements:* Once the system scenarios are sufficiently documented, and each goal is fulfilled by one scenario, the solution-oriented requirements can be modeled. The system still considered as a black box. Use operational, structural, and behavioral requirements models to describe the SUD from the perspective of the context entities. Modeling should focus

on idealized system properties and essential interfaces of the system. Hence, the developed models should be neutral to specific implementation details, but should give closer accounts of *how* the aspects modeled in context, goal, and scenario models are achieved. The modeling of the SUD is a highly iterative and incremental process. It may be possible that, for example, new scenarios (i.e., scenarios missing from the second step) are identified during this step. These newly discovered scenarios may lead to new goals, and so on. This step terminates when no more changes are necessary in the artifacts. Quality requirements are documented relative to the appropriate solution-oriented requirements by means of appropriate annotations. In order to elicit these quality requirements, dedicated analysis steps may be necessary (see Chapter 9).

4.5 References

[Alfaro and Henziger 2001] L. de Alfaro, T. A. Henzinger: Interface automata. In: Proceedings of the 8th European Software Engineering Conference ESEC/FSE-9, 2001.

[Braun et al. 2010] P. Braun, M. Broy, F. Houdek, M. Kirchmayr, M. Müller, B. Penzenstadler, K. Pohl, T. Weyer: Guiding requirements engineering for software-intensive embedded systems in the automotive industry. Computer Science - Research and Development. DOI: 10.1007/s00450-010-0136-y, 2010.

[Cockburn 2001] A. Cockburn: Writing Effective Use Cases. Addison-Wesley, 2001.

[Davis 1993] A. M. Davis: Software Requirements – Objects, Functions, States. 2nd Edition, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[Hammond et al. 2001] J. Hammond, R. Rawlings, A. Hall: Will it work? In: Proceedings of the 5th IEEE International Symposium on Requirements Engineering (RE'01), IEEE Computer Society Press, Los Alamitos, 2001, pp. 102-109.

[ITU 2004] International Telecommunication Union: ITU-T Z.120: Message Sequence Chart (MSC), 2004.

[Jarke and Pohl 1994] M. Jarke, K. Pohl: Requirements engineering in the year 2001 – (Virtually) managing a changing reality. Software Engineering Journal, Vol. 9, No. 6, 1994, pp. 257-266.

[Lamsweerde 2009] A. van Lamsweerde: Requirements Engineering – From System Goals to UML Models to Software Specifications. Wiley, West Sussex, 2009.

[Leveson 2000] N. Leveson: Intent specifications – An approach to building human-centered specifications. IEEE Transactions on Software Engineering, Vol. 26, No. 1, 2000, pp. 15-35.

[Lynch and Tuttle 1989] N. A. Lynch, M. R. Tuttle: An introduction to input/output automata. CWI Quarterly, Vol. 2, 1989, pp. 219-246.

[McMenamin and Palmer 1984] S. M. McMenamin, J. F. Palmer: Essential Systems Analysis. Prentice Hall, London, 1984.

[OMG 2010a] Object Management Group: OMG Systems Modeling Language™ (OMG SysML) Language Specification v1.2. OMG Document Number: formal/2010-06-02.

[OMG 2010b] Object Management Group: OMG Unified Modeling Language™ (OMG UML), Infrastructure v2.3. OMG Document Number: formal/2010-05-03.

[Pohl 2010] K. Pohl: Requirements Engineering – Fundamentals, Principles, Techniques. Springer, Germany, 2010.

[Potts 1995] C. Potts: Using schematic scenarios to understand user needs. In: Proceedings of the ACM Symposium on Designing Interactive Systems – Processes, Practices, Methods and Techniques (DIS'95). ACM, New York, 1995, pp. 247-266.

[Reisig 1991] W. Reisig: Petri nets and algebraic specifications. Theoretical Computer Science, Vol. 80, No 1, 1991, pp. 1-34.

[Weyer 2011] T. Weyer: Kohärenzprüfung von Anforderungsspezifikationen: Ein Ansatz zur Prüfung der Kohärenz von Verhaltensspezifikationen gegen Eigenschaften des operationellen Kontexts. Südwestdeutscher Verlag für Hochschulschriften, 2011.

[Yu 1997] E. Yu: Towards modelling and reasoning support for early-phase requirements engineering. In: Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE'97), IEEE Computer Society Press, Los Alamitos, 1997, pp. 226-235.

4.6 Acknowledgements

The authors would like to thank their former colleagues Dr. Kim Lauenroth (now with adesso AG) and Dr. Ernst Sikora (now with Automotive Safety Technologies GmbH) for their support in early phases of this research.