Extending an IEEE 42010-compliant Viewpoint-based Engineering-Framework for Embedded Systems to Support Variant Management

André Heuer, Tobias Kaufmann, Thorsten Weyer

paluno – The Ruhr Institute for Software Technology
University of Duisburg-Essen
Gerlingstr. 16
45127 Essen, Germany
{andre.heuer,tobias.kaufmann,thorsten.weyer}@paluno.uni-due.de

Abstract. The increasing complexity of today's embedded systems and the increasing demand for higher quality require a comprehensive engineering approach. The model-based engineering approach that has been developed in the project SPES 2020 (Software Platform Embedded Systems) is intended to comprehensively support the development of embedded systems in the future. The approach allows for specifying an embedded system from different viewpoints that are artefact-based and seamlessly integrated. It is compliant with the IEEE Std. 1471 for specifying viewpoints for architectural descriptions. However, the higher demand for individual embedded software necessitates the integration of variant management into the engineering process of an embedded system. A prerequisite for the seamless integration of variant management is the explicit consideration of variability. Variability allows for developing individual software based on a set of common core assets. Yet, variability is a crosscutting concern as it affects all related engineering disciplines and artefacts across the engineering process of an embedded system. Since the IEEE Std. 1471 does not support the documentation of crosscutting aspects, we apply the concept of perspectives to IEEE Std. 1471's successor (IEEE Std. 42010) in order to extend the SPES engineering approach to support continuous variant management.

1 Introduction

Embedded systems bear more and more functionality, must satisfy a growing number of crucial quality demands, and additionally have a higher degree of complexity and inter-system relationships. Key players of the German embedded systems community were involved in the project SPES 2020 (Software Platform Embedded Systems) which was a joined project funded by the German Federal Ministry of Education and Research[1]. SPES 2020 aimed at developing a model-based engineering approach that addresses the challenges mentioned above (cf. [4]).

[1] See http://spes2020.informatik.tu-muenchen.de/

adfa, p. 1, 2011.

The project consortium represented important industrial domains in Germany: automation, automotive, avionics, energy, and healthcare. In the project, the partners from industry and academia jointly developed an artefact-centred, model-based engineering framework for embedded systems that is based on the IEEE Standard 1471 (cf. [9]). This framework is called the SPES Modelling Framework (or short: SPES MF). The SPES MF focusses on the software within an embedded system (cf. [10]) and allows for a seamless engineering of embedded systems, from the requirements to the technical architecture of the system under development (SUD) across multiple abstraction layers (cf. [4]).

Beside the need for seamless model-based engineering, there is a higher demand for the development of different variants of embedded systems. Variant management consists of activities to define variability, to manage variable artefacts, activities to resolve variability and to manage traceability information that are necessary to fulfil these activities (cf. [15]) in each step within the engineering process.

Thereby, variability is defined as the ability to adapt [17], i.e. a development artefact can exist in different shapes at the same time (cf. [15]). The current version of the SPES MF does not support the systematic consideration of variants. As a consequence, concepts and techniques are required for extending the SPES MF to support variant management in the engineering process of an embedded system. A prerequisite for that is the seamless consideration of variability across the engineering artefacts (cf. [3]).

Variability may cause crosscutting changes, for example, in the requirements and the architecture by adapting a system for a specific variant (cf. [14]). A new requirement may impose changes to the architecture. Thus, variability can be seen as a crosscutting concern. Since variability affects all existing viewpoints of the SPES MF, the SPES MF needs to be adapted to deal with such crosscutting concerns. In [10], perspectives are recommended to address crosscutting concerns. We define a Variability perspective for SPES MF that supports the development of different variants of systems in a systematic and comprehensive way.

The paper is structured as follows: Section 2 describes the fundamentals for extending the SPES MF with respect to the consideration of variability in the different engineering artefacts. Section 3 describes our extension of the SPES MF to integrate the Variability perspective in the SPES MF. Section 4 reviews the related work on integrating variability in architectural frameworks. Section 5 gives a conclusion and sketches the future research.

2 Fundamentals

In order to cope not only with the functionality and complexity of a single SUD but also with the variability of a number of similar embedded systems, this section describes the fundamentals to extend the SPES MF for supporting variant management.

2.1 Variant Management in the Engineering of Embedded Systems

Variability is defined as the ability to adapt. Thus, the variability of an embedded system is defined as the ability to adapt the system with regard to a specific context (e.g. context of use, cf. section 1).

It is widely accepted in industry and academia that variability should be documented explicitly in a variability model, which is already a well-proven paradigm in the software product line community (cf., e.g., [5], [12], [15]). This explicit documentation of variability is based on two ontological concepts and their relations. The *variability subject* is defined as a variable item of the real world or a variable property of such an item, e.g. the paint of a car (cf. [15]). The *variability object* is defined as a particular instance of a variability subject, e.g. red paint. A *variant* is a running system that is constituted of a selected set of variability objects. Consequently, in the engineering of variability-intensive embedded systems, *variant management* can be characterized as a process that complements the original engineering process (e.g. requirements engineering, architectural design) by systematically considering variants in each of the engineering disciplines. Performing continuous variant management additionally implies that the relationships of variants are seamlessly documented on a semantic level across the engineering process.

2.2 Viewpoint-Specifications based on IEEE Std. 1471 and IEEE Std. 42010

The IEEE Std. 1471 [9] and its current successor IEEE Std. 42010 [10] introduce a conceptual framework for architectural descriptions (cf. section 1). The key concept of both frameworks is the *architectural viewpoint* (or short: *viewpoint*). To reduce the complexity, the architectural description of a system is typically divided into a number of interrelated views. A viewpoint can be characterized as a structured specification that supports the definition of such a *view* on the system. The specification of a viewpoint consists of the stakeholders' concerns (e.g. specifying the logical architecture) that are addressed by the view together with conventions for creating that view (e.g. the underlying ontology, the ontological relationships to other views, and rules for evaluating the quality of the corresponding views).

Beside the different interrelated views of a system, typically, a system architecture also bears certain crosscutting properties, i.e. properties that have an ontological grounding in each view or an ontological relationship to each one of the views. According to IEEE Std. 42010, architectural models can be shared across multiple views expressing the ontological relationships of the views. This is one possible implementation of the concept of *architectural perspectives* (or short: *perspectives*) introduced by ROZANSKI and WOODS in [16].

2.3 The SPES 2020 Modelling Framework

The SPES MF supports the development of embedded systems by focussing on the following principles (cf. [4]): *distinguishing between problem and solution*, *explicitly considering system decomposition*, *seamless model-based engineering*, *distinguishing*

between logical and technical solutions, and *continuous engineering of crosscutting system properties*. These principles manifest themselves within the SPES MF in two orthogonal dimensions, the SPES viewpoints and the SPES abstraction layers.

The SPES MF Viewpoints. The different stakeholders (e.*g.* requirements engineers, functional analysts, solution architects) in the engineering process of an embedded system have different concerns. Based on the separation of concerns principle, the individual concerns of stakeholders are addressed by certain views that are, in accordance to IEEE Std. 1471, governed by viewpoints in the SPES MF. Each viewpoint addresses certain concerns in the engineering process of an embedded system. The SPES MF differentiates between the following four SPES viewpoints: the *SPES Requirements Viewpoint* addresses the structured documentation and analysis of requirements; the *SPES Functional Viewpoint* addresses the structured documentation and analysis of system functions; the *SPES Logical Viewpoint* addresses structured documentation and analysis of the logical solution, and the *SPES Technical Viewpoint* addresses the structured documentation and analysis of the technical solution.

The SPES MF Abstraction Layers. To reduce the complexity of the engineering process a coarse-grained engineering "problem" is decomposed into a number of fine-grained engineering problems following the strategy of *divide and conquer*, i.e. the composition of the fine-grained solutions is a solution for the coarse-grained engineering problem. Each time, a coarse-grained engineering subject is decomposed into a number of fine-grained engineering subjects; a new abstraction layer is created. Since the number of abstraction layers depends on the properties of the individual engineering context of an embedded system, the SPES MF does not define a certain number of abstraction layers. However the SPES MF provides the mechanism to create new abstraction layers that can be used by engineers to decompose the overall engineering problem to a level of granularity at which the complexity of the fine-grained systems is manageable without the need of performing another step of decomposition.

3 Integrating Variability in the SPES MF

To extend the SPES MF for supporting continuous variant management, firstly, the nature of variability is analysed. Secondly, a general concept for extending the SPES MF is defined and thirdly the specification of the Variability perspective is presented.

3.1 An Insight into the Nature of Variability within the SPES MF Viewpoints

Variability can affect the SPES viewpoints in different ways. Within the SPES Requirements View the requirements of the SUD are specified by using different types of models (e.g. goal models, scenario models). For instance, requirements in terms of system goals (cf. e.g. [8]) are specifying the intention of the stakeholders with regard to the objectives, properties, or use of the system [8]. A variable goal thus represents an objective that may only apply in a specific usage context of the system. Goals can also be contradictory, for example, if a goal of a certain stakeholder excludes a goal of

another stakeholder in a specific usage context of the system. In this situation, these two goals can never be included together in the same variant of a system. Thus, variability in goals may have its origin in variability concerning the stakeholders that have to be considered or in variable intentions of one stakeholder with respect to a different usage context.

In contrast to the Requirements View, in the Technical View hardware components are defined which are implementing specific functions or realizing logical components. Variability on the Technical View could be embodied, for example, by different pins of hardware pieces, or by a different clock speed of a bus. It is obvious that the ontological meaning of variability in the Technical Viewpoint is different from the ontological meaning of variability in the Requirements Viewpoint. In the same way we come to the general conclusion that the ontological meaning of variability is different in all of the SPES viewpoints.

3.2 General Concept for extending the SPES MF for Variant Management

A specific aspect where variability occurs, for instance, in the Requirements View of the SUD is on the ontological level that is different as a variable aspect in the Technical View (❶ in Fig. 1).Variability within the Requirements View can be modelled in an explicit *variability artefact* (i.e. variability model) with a precise ontological relationship to the *engineering artefacts* of the Requirements View (❷ in Fig. 1), whereas variability within the Technical Viewpoint can be documented in an explicit variability model that has a precise ontological relationships to the engineering artefacts of the Technical Viewpoint. This concept can also be applied to other viewpoints and results in distinct variability models for each of the SPES viewpoints.

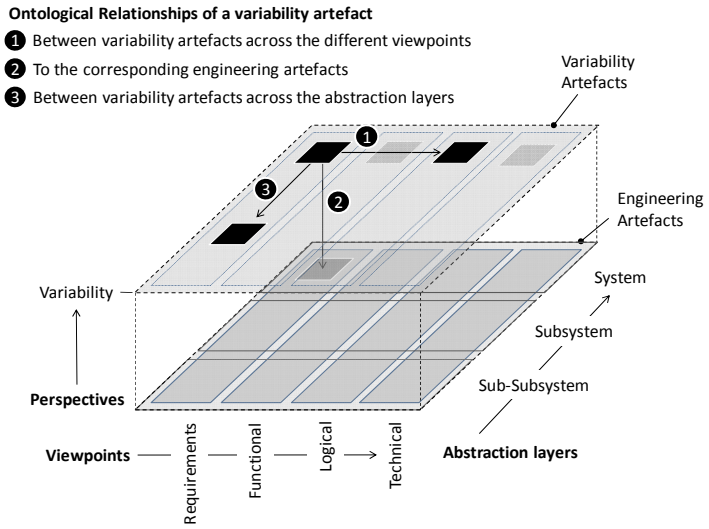

Fig. 1. Variability models in the different viewpoints and their relations

As already mentioned in Section 2.3, today's embedded software is engineered across different abstraction layers based on the SPES MF. Thus, most of the artefact types that are defined based on the underlying ontology of the viewpoints are used on each of the abstraction layers but with a different level of granularity of the engineering subject. On a subsystem layer in Fig. 1, for example, a component diagram models the structure of the SUD. On this level, an interface can be variable. However, on the sub-subsystem layer in Fig. 1, for example, the structure of the different components can also be modelled by a component diagram and interfaces can also be variable and both variable interfaces are related to each other. Additionally, the definition of a variability subject on a higher abstraction layer may lead to different alternatives that impose new variability objects representing different decompositions on a lower abstraction level. Thus, not only artefacts of different types, but also of the same type across different abstraction layers (❸ in Fig. 1) are affected.

The general concept of integrating variability in the SPES MF is also based on the empirical findings and conceptualizations of AMERICA ET AL. [1] as well as THIEL and HEIN [18]. AMERICA ET AL., argue to explicitly document the possible design decisions, by documenting viewpoint relevant variability. Furthermore, they argue that the explicit documentation of choices leads to an increased awareness of such choices, which in turn is beneficial for the stakeholder communication. THIEL and HEIN are interpreting variability as a kind of quality of the architecture of a system in terms of its configurability and modifiability. Variability is materialized in the artefacts by changes or adaptations of specific elements, e.g. interfaces.

3.3 Specifying Crosscutting Aspects conform to IEEE Std. 42010

The SPES MF in its current version does not specify how variant management and thus variability should be addressed in its corresponding viewpoints and abstraction layers. As we already discussed, variant management potentially affects all artefacts and consequently crosscuts all viewpoints of the SPES MF. IEEE Std. 42010 itself provides a mechanism for realizing crosscutting concerns by allowing architectural models to be shared across multiple views and thereby focus on the relevant aspects of a view. Regarding variant management, we belief that this approach is not sufficient, because, as we discussed in section 3.2, the ontological meaning of variability significantly differs in each of the four viewpoints. As a consequence, a shared architectural model would need to be able to represent viewpoint-specific ontological concepts.

ROZANSKI and WOODS [16] also recognized a need for addressing crosscutting aspects fulfilling specific concerns of the majority of system's stakeholders. They are identifying qualities of the architecture (e.g. safety, security) that are affecting all views. To address these qualities, ROZANSKI and WOODS introduced the concept of perspectives, which are defined as [16]: "*[…] a collection of architectural activities, tactics, and guidelines that are used to ensure that a system exhibits its particular set of related properties that require consideration across a number of the system's architectural views*". Perspectives are therefore orthogonal to architectural views. In

[16] a perspective specification template is proposed that addresses quality properties in an IEEE Std. 42010 based specification.

3.4 Specification of the Variability Perspective for the SPES MF

Since variability can be regarded as a quality property and therefore as a crosscutting concern of a system architecture, we extend the SPES MF by following the approach that is described in Section 3.3. To that end we use the template proposed in [16] for specifying the architectural perspective. An excerpt from the specification of the Variability perspective is shown in Table 1.

Table 1. Excerpt from the specification of the Variability perspective

Section	Content
Applicability	Each SPES MF view is affected: • When applying the Variability perspective to the Requirements View, it guides the requirements engineering process of the SUD so that the variability of the requirements can be considered systematically. • When applying the Variability perspective to the Functional View, it guides the functional design for the SUD so that the variability of the system functions can be considered systematically. • When applying the Variability perspective to the Logical View, it guides the design of the logical architecture of the SUD so that the variability within of the logical architecture can be considered systematically. • When applying the Variability perspective to the Technical View, it guides the design of the technical architecture of the SUD so that variability of the technical architecture can be considered systematically. Each SPES MF abstraction layer is affected: • When applying the Variability perspective to an abstraction layer, it guides the systematic engineering of the engineering subjects within that layer so that the variability can be considered across all views of the engineering subject.
Concerns	• *Variability:* the ability of the SUD to be adapted to a different context, e.g. context of usage, technological context, economical context, legal context or organizational context. • *Quality properties of variability:* correctness, completeness, consistent and traceable to its origin and to corresponding engineering artefacts.
Activities	*Steps for applying the Variability perspective to the Requirements View:* • *Identification of variability in the requirements of the SUD:* This step aims at identifying variability in the requirements that is originated by variable context properties. • *Documentation of variability in the requirements of the SUD:* This step aims at documenting the variability in the requirements. • *Analysis of variability in the requirements of the SUD:* This step aims at analysing the variability in the requirements, e.g. with respect to correctness, completeness and consistency. • *Negotiation of variability in the requirements of the SUD:* This step aims at negotiating the variability in the requirements, with the stakeholders of the SUD. • *Validation of the variability in the requirements of the SUD:* This step aims at analysing the variability in the requirements, e.g. with respect to correctness, completeness and consistency. *Steps for applying the Variability perspective to the Functional View:*

	• […] *Steps for applying the Variability perspective to the Logical View:* • […] *Steps for applying the Variability perspective to the Technical View:* • *Identification of variability in the technical architecture of the SUD:* This step aims at identifying variability in the technical architecture that is originated by, for example, variable technical resources (e.g. processors, communication infrastructure) as well as variable sensors or actuators).
Architectural tactics	*Context Analysis and Documentation:* for structured analysis and documentation of the context properties that are the origin of variability • *Orthogonal Variability Modelling:* for explicit documentation of variability and its relationship to engineering artefacts • *Model Checking:* […]
Problems and pitfalls	Problems and pitfalls that may arise: • The increasing complexity of variable artefacts that increases the effort to keep the engineering artefacts consistent. • Complex variability models tend to be ambiguous und confusing, for example, false optional features that are part of every product because of constraints. • […]

4 Related Work

Today, multiple frameworks for designing a system's architecture exist. All these frameworks share the concept of multiple architectural views. In this context, crosscutting concerns are often considered as quality or system properties or non-functional as well as quality requirements of a SUD, which need special consideration when crafting a system's architecture.

In terms of documenting a system's architecture the standards IEEE Std. 1471 [9] and its successor IEEE Std. 42010 [10] provide a conceptual framework for specifying viewpoints governing views (cf. section 2.2). Another approach for documenting a system's architectural views is proposed in "Views and Beyond" [7], which is compliant to IEEE Std. 1471 (cf. [6]).

ZACHMAN proposes a Framework [21], which makes use of six different architectural representations (viewpoints). This framework does not state how crosscutting concerns should be addressed in detail.

The Reference Model of Open Distributed Processing (RM-ODP) [11] proposes five viewpoints, focusing on particular concerns within a system. In addition a set of system properties are defined including quality of service, but it is not addressed how these properties should be encountered during system development.

The rational unified process (RUP) makes use of The 4 + 1 View Model of Architecture, which was introduced in [13]. In RUP four different kinds of non-functional requirements are distinguished, which are subject to an iterative, scenario-based process determining the key drivers of architectural elements. But no explicit guidelines are given how non-functional requirements should be addressed in the architectural design phase. Attribute-Driven Design (ADD) [20] is a method that can be described as an approach for defining a software architecture based on the software's quality

attribute requirements. Essentially, ADD promotes a recursive design process decomposing a SUD making use of the architectural tactics introduced in [2], resulting compliant views to in [7], and consequently explicitly addressing crosscutting concerns.

The TOGAF framework uses the iterative Architecture Development Method (ADM), which contains an analysis of changes etc. in terms of their cross architectural impact. In its current version [18], the TOGAF framework encourages the use of IEEE Std. 42010 in order to craft the necessary viewpoints and views.

ROZANSKI and WOODS [16] take the 4+1 View Model of Architecture as foundation and provide an IEEE Std. 42010 compliant viewpoint catalogue. The stakeholders' requirements as well as the architecture are subject to an iterative architecture definition process. But in contrast to RUP, crosscutting concerns are explicitly addressed in terms of perspectives.

As motivated in subsection 2.3, software intensive embedded systems need special consideration during their engineering. The domain independent model-based engineering methodology of SPES, takes these special needs and challenges into account. In doing so, the IEEE Std. 1471 based viewpoints of the SPES MF are explicitly tailored to the needs of the development of software intensive embedded systems. The above described frameworks are of a more general nature and are consequently not directly applicable in the context of such systems. As motivated in subsection 3.1, variability affects multiple viewpoints and their artefacts. Consequently, it is our firm belief that variability has to be addressed explicitly. Therefore we decided to apply the approach by ROZANSKI and WOODS to the SPES MF in order to address variability explicitly.

5 Conclusion

The specification of the Variability perspective is an essential means for supporting the continuous variant management across the whole engineering process of embedded systems, which are based on the SPES MF. This is done by defining how to seamless integrate, among others, the identification, documentation and analysis of variability and its relationships to the underlying engineering artefacts across the viewpoints and abstraction layers of the SPES MF. In our future work we will apply the extension of the SPES MF for variant management to three industrial case studies (a driver assistance system for vehicles, a mission control software in unmanned aerial vehicles, and a desalination plant) to gain deeper insights concerning the applicability and usefulness for supporting the continuous variant management in the engineering processes of embedded systems.

Acknowledgement

This paper was partially funded by the BMBF project SPES 2020_XTCore under grant 01IS12005C and the DFG project KOPI grant PO 607/4-1.

6 References

1. America, P., Rommes, E., Obbink, J.H.: Multi-view Variation Modeling for Scenario Analysis. In: 5th International Workshop on Software Product Family Engineering (PFE), pp. 44-65, Springer (2003)
2. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. Addison-Wesley, Reading (2003)
3. Broy, M.: Outlook. In: Pohl, K., Hönninger, H., Achatz, R., Broy, M. (eds.) Model-Based Engineering of Embedded Systems – The SPES 2020 Methodology, Springer, Berlin, Heidelberg (2012)
4. Broy, M., Damm, W., Henkler, S., Pohl, K., Vogelsang, A., Weyer, T.: Introduction to the SPES Modeling Framework. In: Pohl, K., Hönninger, H., Achatz, R., Broy, M. (eds.) Model-Based Engineering of Embedded Systems – The SPES 2020 Methodology, Springer, Berlin, Heidelberg (2012)
5. Clements, P., Northrop, L.: Software Product Lines – Practices and Patterns. Addison-Wesley, Boston (2002)
6. Clements, P.: Comparing the SEI's Views and Beyond Approach for Documenting Software Architectures with ANSI-IEEE Std. 1471-2000. Technical Report, Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2005-TN-017 (2005)
7. Clements, P.: Documenting software architectures: views and beyond. Addison-Wesley, Boston (2011)
8. Daun, M., Tenbergen, B., Weyer, T.: Requirements Viewpoint. In: Pohl, K., Hönninger, H., Achatz, R., Broy, M. (eds.) Model-Based Engineering of Embedded Systems – The SPES 2020 Methodology, Springer, Berlin, Heidelberg (2012)
9. IEEE Recommended Practice for Architectural Description of Software Intensive Systems. IEEE Standard 1471-2000 (2000)
10. ISO/IEC/IEEE Systems and Software Engineering – Architecture description. ISO/IEC/IEEE Standard 42010:2011 (2011)
11. ITU-T X.903 | ISO/IEC 10746-3 Information Technology – Open Distributed Processing – Reference Model – Architecture. ISO/IEC Standard 10746-3 (2009)
12. Kang, K.C., Lee, J., Donohoe, P.: Feature-oriented product line engineering. In: IEEE Software. Vol.19 (4), pp. 58- 65 (2002)
13. Kruchten, P.: The 4 + 1 View Model of architecture. In: IEEE Software, Vol. 12 (6), pp. 42-50 (1995)
14. Noda, N., Kishi, T.: Aspect-Oriented Modeling for Variability Management. In: Proceedings of the 12th International Software Product Line Conference. pp. 213-222, IEEE Computer Society (2008)
15. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering: Foundations, Principles and Techniques. Springer, Berlin, Heidelberg (2005)
16. Rozanski, N., Woods, E.: Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives. 2nd edition, Addison-Wesley, Upper Saddle River (2012)
17. The ARTFL Project: Webster´s Revised Unabridged Dictionary (1913+1828). http://machaut.uchicago.edu/?resource=Webster%27s&word=variability&use1913=on, accessed on 19-12-2012
18. The Open Group: TOGAF Version 9.1. 10th new edition, Van Haren Publishing, Zaltbommel (2011)
19. Thiel, S., Hein, A.: Systematic Integration of Variability into Product Line Architecture Design. In: Proceedings of the 2nd International Software Product Lines Conference. pp. 130-153, Springer, Berlin, Heidelberg (2002)
20. Wojcik, R, Bachmann, F, Bass, L., Clements, Paul, Merson, P., Nord, R., Wood, W.: Attribute-Driven Design (ADD), Version 2.0. Technical Report, Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2006-TR-023 (2006)
21. Zachman, J. A.: A Framework for Information Systems Architecture. In: IBM Systems Journal. Vol. 26 (3), pp. 276-292 (1987)