Manfred Broy, Technical University of Munich
Wolfgang Böhm, Technical University of Munich
Bernhard Rumpe, RWTH Aachen University

19

Advanced Systems Engineering

Contribution of the SPES Methodology and Open Research Questions

Advanced systems engineering (ASE) is a new paradigm for agile, efficient, evolutionary, and quality-aware development of complex cyber-physical systems using modern digital technologies and tools. ASE is essentially enabled by smart digital modeling tools for specifying, modeling, testing, simulating, and analyzing the system under development embedded in a coherent and consistent methodology.

The German Federal Ministry of Education and Research (BMBF) projects SPES2020, SPES_XT, and CrESt offer such a methodology and framework for model-based systems engineering (MBSE). The framework provides a comprehensive methodology for MBSE that is independent of tools and modeling languages. The framework also offers a comprehensive set of concrete modeling techniques and activities that build on a formal, mathematical foundation. The SPES framework is based on four principles that are of paramount importance: (1) Functional as well as non-functional requirements fully modeled and understood at system level. (2) Consistent consideration of interfaces at each system level. (3) Decomposition of systems into subsystems and their interfaces. (4) Models for a variety of cross-sectional topics (e.g., variability, safety, dynamics).

W. Böhm et al. (eds.), *Model-Based Engineering of Collaborative Embedded Systems*,
https://doi.org/10.1007/978-3-030-62136-0_19

19.1 Introduction

Cyber-physical systems

Many systems and technical products developed today and in the future are or will be cyber-physical systems. These systems exhibit physical as well as smart, complex, and high-performance functionality, are typically not "stand alone," being instead connected to users and to other systems via digital networks such as the Internet, and their services mutually use and complement each other. It is recognizable that to a certain extent, subsystems, which are created by heuristic procedures, are built into the systems, — for example, by "learning procedures."

Typical for those cyber-physical systems is that they embody software intensively, which enables powerful and connected functionalities that go dramatically beyond what was possible in the past for rather isolated mechatronic systems. The high proportion of software leads to an extensive design space in which the most diverse requirements can be identified. Therefore, the identification of a requirements concept is of particular importance. This also creates extensive potential for innovation, both in terms of purely logical functionality but also very much in human-centered human-machine interaction and automation up to full autonomy.

Cyber-physical systems are characterized by the fact that they usually have mechatronic components, especially sensors and actuators to enable the interaction between physical and software components as well as an interaction of the systems with their environment. These new forms of software enable functionalities through the use of advanced software technology, including artificial intelligence methods, and enable human-centered user interfaces for these systems.

Software as a driving factor

It is particularly noteworthy that today's systems contain an extensive proportion of software for good reasons, as this enables functionalities that were completely out of scope even a few years ago. Due to the strong networking, it is obvious to connect systems with completely different tasks and functionalities — in order to use functionality from other systems, but also to make functionality available for other systems and thus increase the degree of automation and optimization.

19.2 Advanced Systems Engineering

System characteristics

The systems of the future are characterized by the following features:

- ❑ Extensive software components and functionality, which is mainly determined by the software components
- ❑ High degree of networking with other systems for the mutual use of data services
- ❑ Strong integration of software with mechanical and electrical components
- ❑ Comprehensive, dedicated user interfaces
- ❑ High degree of automation up to autonomy
- ❑ Continuous further development — including during operation
- ❑ Complex integration with business software

Characteristics of the ASE approach

These features are also reflected in the required development approaches and determine the characteristics of the advanced systems engineering (ASE) approach. Accordingly, ASE is characterized by the following features:

- ❑ Strong demand for modeling techniques to ensure the correctness of complex functionality and comprehensive tool support
- ❑ Frontloading – shifting efforts towards early phases in development
- ❑ Strong integration of the development processes of the engineering disciplines involved (mainly software engineering, mechanical engineering, electrical engineering) and the tools used; conventional processes such as sequential, discipline-specific development no longer meet the new requirements
- ❑ Strictly systems-centric approach for the holistic integration of the required multidisciplinary design approaches
- ❑ Consistency of development across the family of models, with clear semantic foundations, precisely defined relationships between the models, and development steps systematically develop further models from the elaborated model up to the generation of code and test cases based on well-understood semantic coherence.
- ❑ Equal support of a top-down and bottom-up approach via the consistency of the transitions between the models.
- ❑ Close interlinking of the data-driven and model-driven approach through harmonization of the component and interface concept.
- ❑ Intensive use of software tools for all phases of development, consistent artifact orientation, virtual development by creating suitable digital artifacts, automation of the development process

through simulation, generation and automated deduction and quality verification

- ❑ Merging of development and operation (continuous development and delivery, DevOps, agility)
- ❑ Use of development models for further evolution and during operation (from system model to digital twin)
- ❑ New types of cost structures, higher development costs in relation to lower production costs due to the often dramatically higher variability
- ❑ Intensive integration of new forms of software and the resulting possibility of adding new and modified functionality even during operation of the systems leads to new types of business models

It is evident that these points interact with, complement, and reinforce each other.

19.3 MBSE as an Essential Basis

Formal system model

The approach pursued by MBSE is clearly distinguished from the document-centered, manual approach that is still widely used today. Objectives, functions, components, interfaces, or quality properties that a system fulfils or provides are described by explicit model elements based on well-defined and well-understood concepts of the domain. A number of modeling concepts are used in model-based development. The concepts are selected in such a way that they capture the essential system properties clearly and precisely. A separate theory can be specified for each of these modeling concepts. The same applies to the description of the relationship between the different modeling concepts used. This has the advantage that users trained in the approach (similar to programming languages) are familiar with the concepts and know which models they have to apply to certain questions. Engineers thus also know the basic problems they have to deal with in order to create the models in a goal-oriented way and use them in system analysis or synthesis. Model-based development is much more than just drawing or setting up models; it also includes the comprehensive use of an elaborated modeling approach.

Pre-built model types, based on a scientific foundation, guarantee properties such as compositionality, which clearly defines the integration of subsystems described by models (such as communication via an interface) and reuse. The models must be coordinated in terms of content and engineers must understand exactly how the different modeling approaches interact. One important point is the semantic coherence across model boundaries and the boundaries of modeling languages, which ensures that a comprehensive model of the system is created. A system description is then no longer this vaguely informal structure of documents, but rather an interwoven network of standardized models that form a common whole. An instance of the system model, which in this form is then consistently stored in a central model repository ("single point of truth"), is managed. Stakeholders have different views of this central system model that are tailored precisely to their respective roles in the product life cycle (for example, function developer, architect, service). This avoids undetected inconsistencies and, in particular, simplifies the ability to change the models, thus reducing a significant cost driver.

The transition from textual descriptions in natural language to models also has the advantage of reducing ambiguities, making consistency and completeness verifiable, and improving communication between stakeholders. The more formal the model used, the less ambiguity there is in the description. More importantly, formal models enable automatic analyses—for example, to check the interaction of the individual components—and they also allow the use of generators to generate parts of other models and artifacts (such as code or test cases) from elaborated system models.

This shows that model-based development constitutes *one,* perhaps *the* key to advanced systems engineering with a high level of tool support.

Tool-supported development

Another important point here is the possibility of tool-supported development of systems. Here, the software is of particular interest in two respects: on the one hand, the development of systems in their inevitable complexity will be supported in a way that is indispensable to advance such systems in general; on the other hand, supported development requires comprehensive, systematic modeling and thus a virtual registration of the systems. This means that these models can be used as digital twins for the operation of the systems and thus ensure even more extended functionalities. The software optimizes itself during development, so to speak.

Aspects of MBSE

Three aspects of MBSE must be considered separately, but nevertheless skillfully coordinated with each other:

Methodology: A system model, which in turn consists of a multitude of model types, is itself a complex artifact that cannot be created effectively and efficiently without an underlying science-based methodology. Therefore, an MBSE methodology includes the definition of the relevant model types and their relationships. Furthermore, it defines views of the system model, which structure the complex overall model into several, less complex models that are adapted to the given development situation. Examples of views include functional and logical or technical architecture views. An MBSE methodology also describes possibilities for analysis and generation of the specific models. The degree of formalization of the models defined in the system model determines the degree of automation of analyses and model generation. A high degree of automation, which of course must also be supported by the tools used, allows in turn an iterative and agile development process, such as in pure software development.

Modeling language: The modeling language defines the syntax and semantics used to describe the models of the methodology in concrete terms — for example, which textual or graphical notations are allowed (syntax). The semantics of a modeling language defines the meaning of these notations. The problem here is that many of the common modeling languages (such as SysML) have at best a loosely defined semantics. This causes problems similar to those of natural language descriptions.

Tools: The methodology and language must be supported by appropriate tools in order to make efficient use of the possibilities offered by models. It is also crucial that the tool chains used are compatible with each other and that the tools used support the chosen methodology and the modeling language both syntactically and semantically and with a high degree of automation.

19.4 The Integrated Approach of SPES and SPES_XT

Principles of the SPES modeling framework

In the BMBF project SPES2020 [Pohl et al. 2012] and its successors SPES_XT [Pohl et al. 2016] and CrESt, a methodology and framework for MBSE were developed that allow efficient model-based development of embedded systems. The SPES framework provides a comprehensive methodology for MBSE that is independent of tools and modeling languages. The framework also offers a comprehensive set of concrete modeling techniques and activities that build on a formal, mathematical foundation. The SPES framework is based on four principles of paramount importance:

- ❑ Functional as well as non-functional requirements fully modeled at system level using appropriate abstractions (views)
- ❑ Consistent consideration of interfaces at each level

- Decomposition of the interface behavior and the description of systems via subsystems and components at different levels of granularity
- Definition of models based on the above principles for a variety of cross-sectional topics (variability, safety, etc.) and analysis options

System model in SPES

A system model in the SPES approach is a conceptual ("generic") model for the description of systems and their properties, consisting of:

- Models for the operational context that influences or is influenced by the system at runtime
- Models of the interface that clearly delimit the system from its operational context
- A behavior of the system that can be observed at the interface
- Models of the internal structure of the system implemented by state machines or by interrelated and communicating subsystems (architecture) to which the SPES framework can be recursively applied

Universal interface concept

The core of the methodology is the *universal interface concept*, which defines interfaces for all elements, each consisting of the interface syntax and a description of the behavior observable at the element boundary. Requirements, functions, and logical or technical components are thus described via the interface and are connected to each other via their interfaces. The interface concept provides the basic decomposition and modularity.

Views and crosscutting topics

Views [IEEE42010 2011] in the SPES framework are the requirements view, functional view, logical view, and technical view. They decompose the system into the logical or technical components involved. Crosscutting topics supplement the models of the views accordingly. For example, this allows aspects of the functional safety of systems to be described and analyzed. The SPES framework is open to the addition of new views, such as a geometric view.

Levels of granularity

In order to make the complexity of the system and the associated development process manageable, relevant architectural components are considered as independent (sub)systems according to the principle of "divide and rule." For these systems, models and views are created according to the SPES approach. This creates predefined views for the system and its subsystems with matching levels of granularity. The modeling of the system at the different levels of granularity determines the subject of the discourse (scope) and is an important tool in model-based development to reduce system complexity and to make the development process manageable.

Mathematical foundation

The SPES MBSE methodology follows a strict system-centric approach that specifies a system at several levels of granularity. At the highest level of granularity, there are always the models that represent the system under consideration as a whole. At (varying numbers of) further levels of granularity, increasingly fine subsystems are successively considered, and further details are modeled. Although "top-down" is the basic principle, iterative, agile, and evolutionary processes are also supported. The mathematical model FOCUS, on which the SPES framework is based, ensures the consistency of the models of systems and subsystems. Levels of granularity help to (1) control the complexity of the system under consideration, (2) perform checks on the system at different levels of complexity, (3) distribute development tasks—for example, to suppliers—and (4) reuse individual models several times. Since the principle of granularity levels is based only on the interface concept, the mechanism allows the integration of the different engineering disciplines (mechanical engineering, electrical engineering, software engineering). As long as the interface concept is realized, the methods, processes, or tools used to develop the subsystems at the lower levels of granularity are irrelevant.

Consistency of the models

Besides abstraction and granularity, consistency is an important feature of the models in the SPES framework. We distinguish between horizontal consistency and vertical consistency. Two models are horizontally consistent if they belong to different views of the same system (i.e., are within one level of granularity) and do not represent contradictory properties of the system under consideration. Two models are vertically consistent if they belong to one view at different levels of granularity and do not represent contradictory properties of the system under consideration with regard to the specific view.

Agile and iterative development

The SPES framework does not specify the order in which the different models should be created for the views. Thus, the SPES method can be used to implement top-down as well as bottom-up approaches and iterative or incremental development and even evolution. As mentioned above, the mechanism of granularity levels allows the integration of different approaches and development tools, as typically required for mechatronic systems. Since the formal basis of the SPES methodology also supports under-specification, it is possible to extend and successively refine the models iteratively step by step. This means in particular that the model-based approach does not contradict but rather supplements the basic principles of agile development. Techniques such as "continuous integration" can also be used in a purposeful manner. It should be emphasized that this form of an agile approach is not just code-centric but also model-centric.

Extension towards networks of systems

In the CrESt project, the SPES framework was extended to support collaboration and dynamics (formation of system networks at runtime) in systems. The existing viewpoint structure was essentially retained, but the models contained within the structure were extended by additional model types and information.

19.5 Methodological Extensions: From SPES to ASE

SPES contribution to ASE

Advanced systems engineering (ASE) is definitely a new paradigm for agile, efficient, evolutionary, and quality-aware development of complex cyber-physical systems using modern digital technologies and tools. As said earlier, ASE is essentially enabled by smart digital modeling tools for specifying, modeling, testing, simulating, and analyzing the system under development embedded in a coherent and consistent methodology.

Model-based systems engineering is thus a core element of ASE and the SPES methodology, as a fully model-based approach, therefore provides an excellent basis for ASE. In particular, the SPES methodology includes:

- ❑ Consistent models that cover the entire product development process
- ❑ A variety of modeling techniques to ensure and analyze the correctness of complex functionality
- ❑ Modularity and decomposition, which allow reuse of model elements at all levels
- ❑ Consistent architecture views and executable model elements, which allow functional prototypes and automated analyses in early phases of the development process (frontloading)
- ❑ Strict system-centric approach to support the necessary multidisciplinary design approaches
- ❑ Integration of the development processes of the engineering disciplines involved (computer science, mechanical engineering, electrical engineering) and the tools used there via the concept of granularity levels
- ❑ Extensive models especially for software engineering and strong integration of software with mechanical and electrical components

- Extension of the SPES framework towards aspects such as dynamic networking and collaboration of systems at runtime in the CrESt project; for this purpose, a number of additional crosscutting topics were defined, and the models of the existing viewpoints were supplemented accordingly

New methodological and crosscutting issues would be, for example:

- Extension of the predominantly discrete models to analog models; integration of control engineering approaches — keyword "interdisciplinary modeling"
- Integration of novel methods for the generation of subsystems and their behavior through big data and machine learning
- Integration of security models for safety and security into model-driven development with a focus on certification
- Consideration of digital twins as part of the overall system to be developed
- Quality assurance at runtime
- System qualification and certification
- Dedicated user interfaces

Further development towards ASE

Up to now, the development of the SPES framework has focused exclusively on the product development process. At the same time, SPES offers the possibility to add new viewpoints to the already existing viewpoints or to extend the existing viewpoints via additional crosscutting topics and integrate them into the framework. A further development towards ASE should therefore take into account extensions towards the entire product life cycle, including models and extensions for market and business models as well as system operation and service models.

SysML as a modeling language

The models, methodology, and techniques developed in the SPES, SPES_XT, and CrESt projects were deliberately written independently of a specific modeling language in order to ensure the greatest possible range of application. In industrial practice, especially in small and medium-sized enterprises, it has been shown that almost all MBSE implementation projects rely on SysML as a modeling language, despite all the open questions and shortcomings associated with it. The reason for this is the spread of SysML in companies as well as the support of SysML in many MBSE tools available in practice. Due to the spread and acceptance of SysML, it must be explicitly supported as a modeling language both syntactically and semantically with the SPES methodology. A research project based on the SPES framework is planned that will break down the current barriers to the industrial introduction of MBSE and thus pave the way for a broad industrial adoption of the SPES methodology based on common language syntax and pragmatic tools.

Artificial intelligence

Parts of future systems will be determined by the use of techniques such as machine learning (ML) or, more generally, artificial intelligence (AI). Integrating AI components into embedded systems leverages the considerable potential of current and future AI technologies in embedded systems. Their use enables future embedded systems to suitably process the constantly growing volume of information resulting from digitalization and to adapt to changing conditions and to the knowledge gained from the data at runtime. In order to be able to develop such systems efficiently, the explicit modeling methods available must be extended by implicitly learned modeling techniques. In principle, the approach presented here is already suitable for systems that have AI components. The universal interface concept of the SPES framework provides a sustainable basis for this. However, it has to be considered that the behavior of such systems is subject to a certain variability during runtime — for example, if the component continues to learn during runtime.

Quality properties

One central challenge for the integration of AI methods into embedded systems is therefore the guarantee (verifiability) of the essential functionality and quality properties of the systems — and this despite the fact that system components cannot be completely specified and are often non-deterministic or even dynamically adapting due to adaptations of the systems at runtime that could not be foreseen at development time.

19.6 Conclusion

ASE requires a clean scientific foundation and a consistent integration of software development and system development methods when designing software-intensive cyber-physical systems. Central to advanced systems engineering is the use of digital techniques in both the product and the development process and the exploitation of the synergies between them. The preliminary work in the area of model-based development of software-intensive systems offers an ideal entry point. Nothing less than a paradigm shift from the engineering of mechanical machines to the integrated engineering of networked, information-centric mechanical systems must be mastered.

19.7 Literature

[Broy 2010] M. Broy: A Logical Basis for Component-Oriented Software and Systems Engineering. In: The Computer Journal, Vol. 53, No. 10, 2010, pp. 1758-1782.

[Broy and Rumpe 2007] M. Broy, B. Rumpe: Modulare hierarchische Modellierung als Grundlage der Software- und Systementwicklung. In: Informatik-Spektrum. Springer Verlag, Band 30, Heft 1, 2007 (available in German only).

[Broy and Stølen 2001] M. Broy, K. Stølen: Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement, Springer, 2001.

[Broy et al. 2007] M. Broy, M. L. Crane, J. Dingel, A. Hartmann, B. Rumpe, B. Selic: UML 2 Semantics Symposium: Formal Semantics for UML. In: Models in Software Engineering. Workshops and Symposia at Models 2006. Genoa, LNCS 4364, Springer, 2007.

[Broy et al. 2020] M. Broy, W. Böhm, M. Junker, A. Vogelsang, S. Voss: Praxisnahe Einführung von MBSE – Vorgehen und Lessons Learnt, White Paper, fortiss GmbH, 2020 (available in German only).

[IEEE42010 2011] ISO/IEC/IEEE 42010:2011: Systems and Software Engineering — Architecture Description. International Organization for Standardization, 2011.

[Pohl et al. 2012] K. Pohl, H. Hönninger, R. Achatz, M. Broy: Model-Based Engineering of Embedded Systems, Springer, 2012.

[Pohl et al. 2016] K. Pohl, M. Broy, M. Daembkes, H. Hönninger: Advanced Model-Based Engineering of Embedded Systems, Extensions of the SPES 2020 Methodology, Springer, 2016.