Strategic Considerations when Introducing Model Based Systems Engineering

Christoph Bräuchle[1], Prof. Dr. Manfred Broy[2], Dominik Rüchardt[3],

[1]Parametric Technology GmbH, Posener Str. 1, D-71065 Sindelfingen, cbraeuchle@ptc.com

[2]Technical University of Munich Institute for Computer Science
Boltzmannstraße 3, D-85748 Garching bei München

[3]Parametric Technology GmbH, Edisonstr. 8, D-85716 Unterschleissheim

The increasing technical complexity of innovative products demands new development approaches that coordinate activities across the involved engineering disciplines such as mechanical, electrical and software engineering. Systems Engineering and specifically model-based Systems Engineering (MBSE) addresses many challenges that arise from the multi-disciplinary and high-complexity nature of modern product development. However it also requires a cultural change in the organization as new competences have to be established, new processes must be described, conventional development methodologies need to be refined and appropriate tool-support has to be ensured. This paper explores some of the key challenges and solution approaches in the engineering of complex embedded systems and introduces a new improvement model to guide the change process towards a modern systems engineering oriented development organization.

Engineering of innovative products

The past decade has shown a significant shift of product innovation towards software, which is embedded in complex systems. This trend has started in the 1980s with the introduction of fly-by-wire in the Airbus A320 and safety controls in premium automobiles such as ABS in the Mercedes S-Class. Software-controlled functionality was increasingly adopted to high-volume products as electronic components like processors and memory became inexpensive – adding to the technical complexity of these products. Over ninety percent of today's automobile innovations are enabled by software – often without being noticed by the end-user. For example, most drivers of modern cars are not aware that their throttle control is just a sensor that sends an electronic signal to the engine control unit where software algorithms determine the optimal throttle angle.

Another important paradigm has changed since the turn of the millennium: ubiquitous mobile connectivity has become a commodity and thus cyber-physical systems are finding their way into all fields of technology. This allows for new types of functionality that are not only implemented within one embedded system, but are made possible through cross-system communication. As a result more and more functionalities are provided by a system-of-systems. Examples can be found in the health-care industry where home-monitoring services for cardio or diabetes therapy can be connected with implanted devices such as insulin pumps or pacemakers and allow for enhanced and immediate interaction with the therapist. Also the automobile industry provides modern telematics, diagnostic and even update functionality through cloud-based solutions that are best described as a system-of-systems.

So we see two directions of increased technical complexity: embedded software controls a continually increasing part of system functionality, and on the other hand connected systems allow for additional functionality through cloud-based services or through communication with other, external systems. In this trend we observe that the end-users do not want to be confronted with this increasing technical complexity. Quite the contrary, ease-of-use is an important criterion for end-user acceptance and thus market success.

Therefore a key prerequisite to achieve this market success with innovative, yet easy-to-use functionality is the ability to master the interaction between mechanical and electronic

components with integrated software and connected systems at any point in the product lifecycle. This is probably the most important challenge in safety-critical and high-complexity product development. Holistic Systems Engineering approaches with model-based engineering practices using formal notations like SysML have been researched and recommended as the way forward in projects like SPES 2020[1], SPES XT or mecPro[2][2]. It helps the engineers to understand the structure of embedded systems with their mechanical, electronic and software components, to comprehend the extent and nature of interactions between these components and define the interoperability with people and connected systems in their operational context.

In the following section of this paper we describe some key characteristics of such a holistic Systems Engineering approach with a focus on model-based development as today's best practice to successfully design software-intensive embedded systems and cyber-physical systems. Since such an approach introduces new practices, methods and also tools, new competences and even roles in the organization must be established. It is therefore important to consider not only the processes, methods and tools, but also the people working in this environment, to allow them a stepwise adoption of the new practices and tools and to provide appropriate training. The third section of this paper describes a framework that can be adopted for such a stepwise introduction of Systems Engineering.

Key development activity requirements and architecture

Complex systems typically consist of a large number of subsystems and components. They are also multifunctional, providing a family of functions and functional features. An important aspect of a model-based architecture the description of both, the structural and the functional decomposition of a system. Another good practice is the distinction between a logical structure, where functions or functional contributions can be allocated to logical building blocks, and a technical structure where the logical building blocks are distributed to actual technical elements such as various control-units, sensors and actuators.

The SPES 2020 methodology therefore recommends four viewpoints in a model-based architecture[3], which are also reflected in Figure 1. The requirements viewpoint describes the system from the view of stakeholders interacting with the system using structured text and semi-formal notations as well as graphical notations such as use-case diagrams. The functional viewpoint provides a model-based behavior specification of the system starting with functions that can be experienced by the end-user and refining these with functional contributions on subsystem and component level. The logical viewpoint describes the internal logical structure and the behavior of the logical building blocks focusing on both, their internal behavior as well as their interactions. At this point the technical implementation – for example actual signals, or how they are transmitted – is not yet specified. The technical viewpoint finally models the system in terms of resources such as processing time (software), available memory (electronics) or even physical limitations like a maximum torque (mechanics). Therefore this viewpoint can be used not only to distribute functions or functional contributions by logical components to physical parts or assemblies, but also to coordinate the allocation of product design aspects to specific engineering disciplines.

We have established the composite nature of complex systems, which requires consistent decomposition into subsystems and components. The best practice for this continuous refinement from system to component level is the definition of different abstraction levels. The actual number of abstraction levels varies in most development projects. In this paper we use three layers, which form a recursive pattern – i.e. a subsystem can be considered as a system in itself and further decomposed.

---

[1] Prof. Dr. K. Pohl, H. Hönniger, Dr. R. Achatz, Prof. Dr. M. Broy (Eds.): Model-Based Engineering of Embedded Systems, the SPES 2020 Methodology, Springer.com

[2] www.mecpro.de

[3] Dr. W. Böhm et al: Bridging the gap between Systems and Software Engineering by Using the SPES Modeling Framework as a General Systems Engineering Philosophy, CSER 2014

1. The operational level describes the system in its operational context interacting with stakeholders and other systems. It uses an operational model to fully understand these dependencies and interactions, ensure a complete requirements specification and full test coverage.
2. Based on this understanding of the system in its operational context, it is possible to derive the actual system specification using a model-based system architecture as central source of information. The technical viewpoint of this model can then be used to decompose the system.
3. The result of this decomposition are specifications on component level. Most systems can be decomposed into subsystems, which can be further decomposed, but as this principle forms a recursive pattern, we just describe the component level.

Figure 1 visualizes these abstraction layers and shows how a model-based architecture should be linked with textual requirements specifications and also test assets like test plans. Although a complete model-based architecture can be an exhaustive specification of a system, we still see the need for textual requirements specifications as contractual basis in development projects with external partners and suppliers. Whether this will change in the future or not, it is currently important to have a well-structured requirements specification that is consistent with the model-based architecture and directly linked with test cases that provide evidence that the design and implementation of the system fully satisfies the requirements.

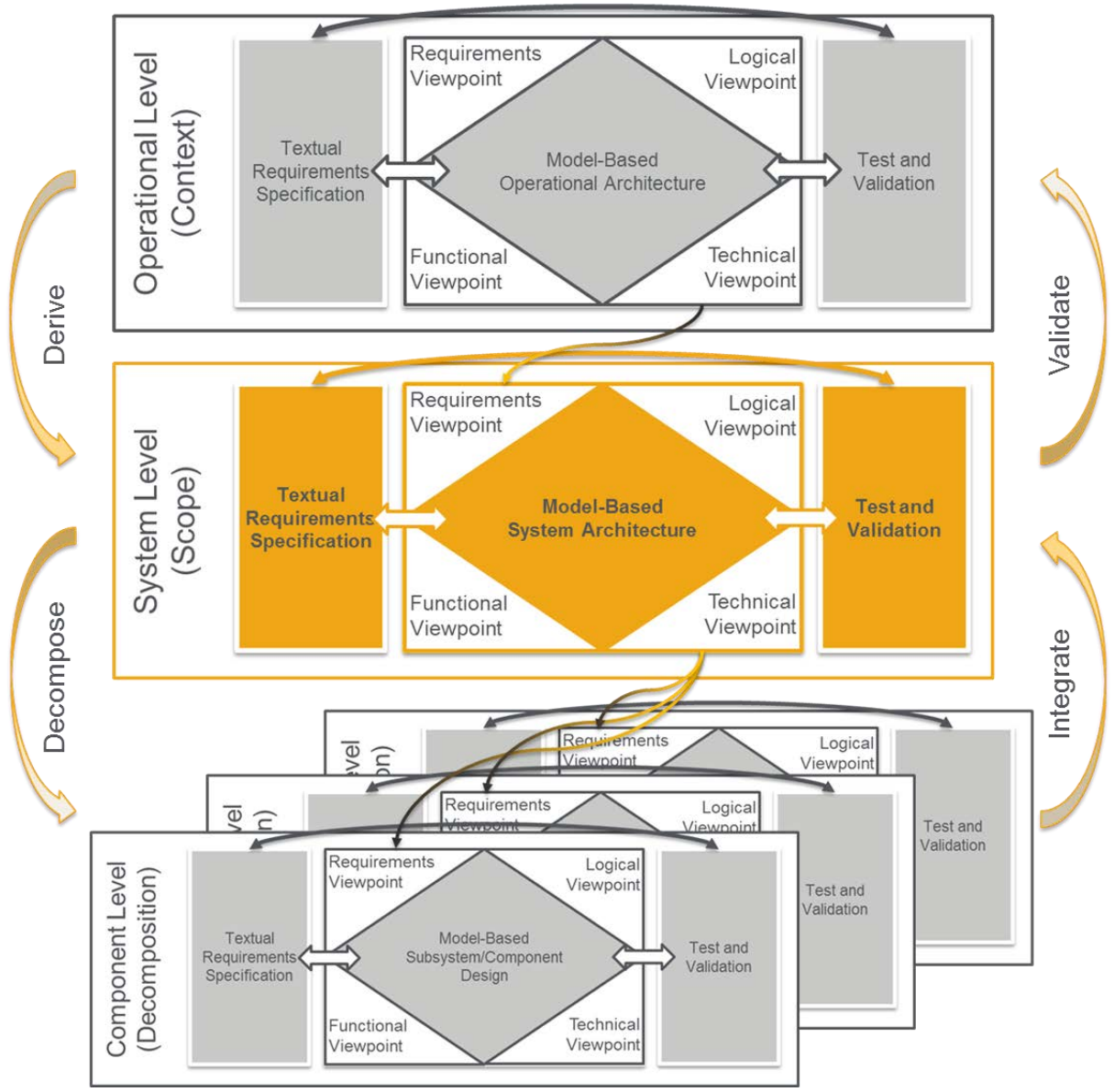

Figure 1: Abstraction layers to specify complex systems

So a key concept in this holistic systems engineering approach is the utilization of a model-based architecture, which is structured into the four viewpoints across multiple abstraction layers, but embedded in a framework abstraction layers and requirements management and test-engineering. This approach helps addressing several challenges in complex systems engineering:

- Multi-functionality can be mastered by the continuous refinement of functional contributions in the functional viewpoint of a model across multiple abstraction layers
- The need for multi-model engineering is addressed by using one central system architecture, which serves as single point for consistent decomposition into multiple component models. Each of these can be designed in the best suited model – e.g. software controls in a software modeling tool, mechanical design in CAD, etc.
- Usability design of human machine interfaces is already addressed on the operational level and all relevant aspects become subsequently covered on the lower abstraction layers.
- System families can be defined by identifying commonalities in the operational context.
- Systematic reuse is facilitated through the modular design with a focus on interfaces and component interactions – this allows for systematic component reuse
- Challenges resulting from the distributed nature of systems can be specifically addressed in the respective abstraction layer
    - Concurrency
    - Real-time communication
    - Adaptation
    - Autonomy
- Safety aspects are covered by seamless traceability and the direct connection to test management, which is a solid basis for documenting the safety-cases.
- Security can be ensured as such an architecture helps identifying all communication interfaces and determining the appropriate security measures (e.g., authentication, encryption, etc.)
- Reliability aspects can be included in the design starting on the operational level
- Interoperability is also considered on all levels and a well-structured, model-based architecture helps identifying applicable standards to ensure interoperability with other systems or third-party components.

Basic principles to observe and apply in an engineering process following this holistic systems engineering approach are accurate requirements capturing and documentation, careful design covering functional, logical, and technical structures and comprehensive elaboration of quality attributes for system. When developing low or medium complexity systems, these principles can be ensured by well-educated engineers using manual review processes. However, safety-critical high-complexity systems engineering requires appropriate tool-support.

Tool support

State-of-the-art modeling tools not only provide the ability to design complex systems across multiple abstraction levels, but also allow for early simulation of the system behavior based on its functional design – e.g., using state machines and activity diagrams. However, considering the entire systems engineering approach we have introduced before it becomes clear that a universal "all-in-one" solution with a monolithic repository is not flexible enough to react to specific and fast changing requirements. Therefore toolchains are required that are based on an open architecture, which allows for flexible integration of domain- and discipline-specific tools.

A tool-chain that can support modern systems engineering can be characterized by these principles:

- Comprehensive support for all activities (model/artifact building, analysis, validation and verification generation management)
- Work products like models, diagrams, textual specifications and tests are authored and managed with support of the tool suite

- Model development and transformations from one viewpoint to the next – i.e. requirements to functional viewpoint, functional to logical and logical to technical – are seamlessly supported by the tools
- Work products are stored consistently in a data back-bone
- Support for concurrent and also distributed development
- Role-based and project-based access rights management

From state-of-practice to state-of-the-art

Migration approach

The path to a comprehensive development process, which fully embraces a structured description of systems, from an existing development practice requires significant effort. Companies making this shift face various process, tool, and method changes. Apart from understanding the future of systems engineering, it is also important to have a viable migration strategy transitioning current development practices to a forward-looking systems engineering environment.

The most efficient approach to migration:

- Assess the current development practice and the maturity of the existing systems engineering environment
- Understand long-term goals and define relevant characteristics required of the future systems engineering environment
- Define incremental steps towards the envisioned goal, where each step should yield tangible benefits
- Plan and execute the necessary activities to reach each incremental step

Following this approach is vital for the success of a project to introduce systems engineering because the people are the most important element in a holistic systems engineering environment. Without a clear strategy that covers not only technical and organizational aspects but also ensures transparent communication and builds competences for the new roles, such a migration approach has a high risk of failure.

Creating a framework for step-by-step migrations

To support a migration initiative we propose to follow a framework that is inspired by similar approved approaches, like SEI's SE-CMM. The main objective of this framework is to provide a lean and actionable structure that is easily adapted, augmented, or further extended to suit the needs of each individual development organization. The intention of this framework is to provide guidance on assessing the current state of practice in systems engineering and based on this assessment lay out a transition plan to the next level of maturity in systems engineering:

- Prepare a roadmap to improve the systems engineering ecosystem in order to achieve higher efficiency, quality, and innovation
- Determine focus areas where important challenges must be mastered to reach a higher maturity level
- Identify and agree on measurable goals in pilot projects and over the entire migration process to realize and track successes

A shown in Figure 2, the base framework first describes four cornerstones in a systems engineering ecosystem: Requirements and validation, model-based architecture design, cross-discipline coordination and product line engineering. It then highlights one extended area of systems engineering. We have described integrated simulations as a frequently needed process area but each engineering organization may have more of these extended areas, such as e.g., code-generation or safety engineering. Finally the framework describes IT considerations that are important for a well-defined systems engineering environment. For each area it shows three key characteristics, one process-related, one method-related, and one tool-related.

	Systems Engineering Cornerstones				*Extended SE Areas*	*IT Considerations*
	Requirements and Validation	**Model-based Architecture Design**	**Cross-Discipline Coordination**	**Product Line Engineering**	**Integrated Simulations**	**SE Environment Infrastructure**
Process	Complete and consistent requirements capturing, decomposition and test specification	Model-based system architecture as guiding source of information in the engineering process	Seamless Change and Configuration Management across the entire product lifecycle	Process-driven strategic reuse and design for variability	Early Simulations embedded in Engineering Process	Distributed and Global Engineering Support
Method	Structured requirements engineering approach with full end-to-end traceability	System design following a methodical approach using multiple viewpoints	Architecture-based discipline-allocation, cross-discipline impact analysis	Standards-based description of variability (e.g. using OVM) and tracking of dependencies	Early validation of system behavioral design based on functional viewpoint modeling	Harmonized Project Structure and Information Model
Tool	Integrated management of requirements and validation data	Open too for model-based system design that allows for integration of discipline-specific modeling tools (SW, CAD, EE)	user interaction and data flow across PLM and authoring tools for MBSE, requirements, test and software engineering	Variant model to manage variability of requirements, tests, models, etc.	Integrated co-simulation platform, simulation interfaces	Repository Structure, Access Rights Management, Archiving

Figure 2: Base framework for maturity evaluation and migration strategy.

Based on these cornerstones and the extended process areas the framework then considers five maturity levels from “inconsistent or ad-hoc” to “optimizing” – as shown in figure 3. It suggests attributes to determine the current maturity level or set goals in a migration initiative to increase systems engineering maturity. These attributes are described again for each aspect: process-, method- and tool-related. Figure 3 also highlights the required competences and finally indicates the area in which the most benefits or improvements can be expected after successful transition to the next level. Of course, figure 3 only focuses on one cornerstone “Model-based Architecture Design”. Similar descriptions exist for each cornerstone in the proposed framework.

So this framework can be used as a systems engineering and development benchmark, as well as a guideline for active migration and – most importantly – the layout of a stepwise training and education program for the engineering team.

An efficient approach to such an active migration can start with the consolidation and harmonization of each relevant area on one maturity level. This is followed by a step-by-step improvement program, where each element provides tangible added value in terms of quality, profitability, or time-to-market. The use of five maturity levels is a proven strategy to determine achievable improvement steps.

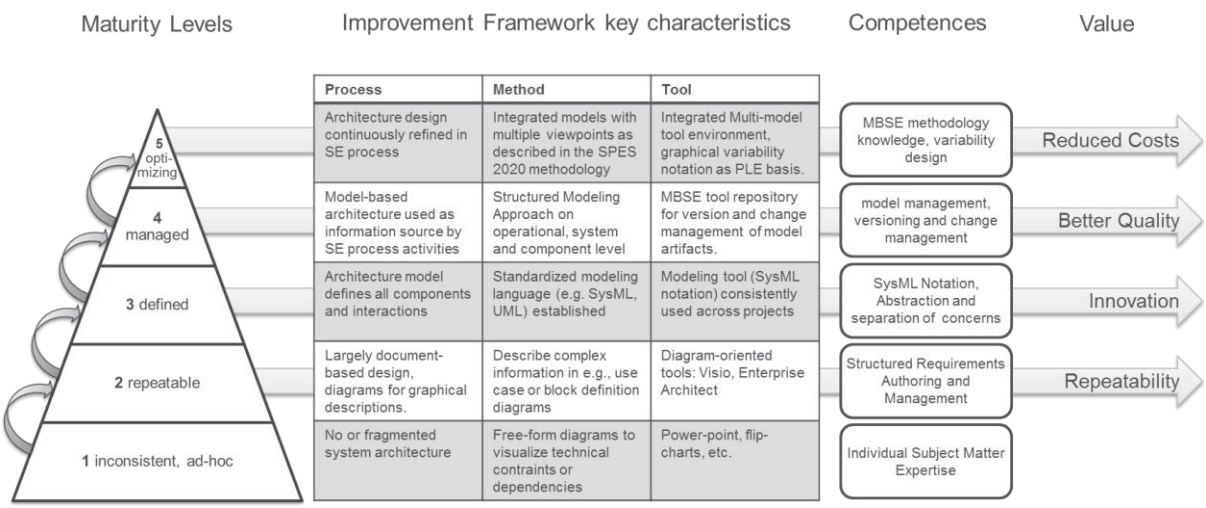

Figure 3: Focus on the "Model-based Architecture Design" Cornerstone – Maturity levels, key characteristics and required competences

Conclusion

This strategy allows the transformation to a modern and trend-setting engineering organization to occur without the risk of disruptions and typical overhead-work for training and extensive data migration. Furthermore, it supports a flexible timeline during the migration initiative as it minimizes the dependencies between the individual areas. Finally, it ensures quick and noticeable value wins as the specific challenges in the engineering process of smart, connected systems are addressed step by step.