Fraunhofer
IESE

WP2/D2.1: Systems Engineering Study: Challenges and Best Practices

Authors:
Jens Heidrich
Binish Tanveer
Rolf van Lengen
Thomas Kleinberger
Liubov Gorodilova
Thomas Kuhn
Martin Becker
Thomas Bauer
Andreas Morgenstern

Table of Contents

Management Summary

This booklet discusses the general trend towards Systems Engineering, summarizes existing studies in the field, and highlights results from an interview series about challenges and best practices in the area of Systems Engineering across innovative companies in the German-speaking region. Furthermore, selected best practices are explained and cases of companies applying these practices are summarized. The study was performed by Fraunhofer IESE in Germany and was sponsored by the Ministry of Economy, Trade and Industry in Japan.

Part 1 The amount of software in formerly largely hardware-dominated products has continuously increased over time. Software is perceived as an enabler of new, innovative services and business models in all sectors of industry and society. Systems Engineering is an interdisciplinary approach that considers both the business and the technical needs of all customers. Being able to establish appropriate Systems Engineering practices in the organization is crucial for staying competitive and for developing innovative products on time, within budget, and with a high level of quality.

Part 2 Our goal was to collect the state of the practice regarding Systems Engineering in the German-speaking region, focusing on challenges and solution approaches in terms of best practices (work processes, methods, and tools). The scope of the study was on Systems Engineering practices across different domains and was not specialized to any single domain. Even though we found some already existing surveys and studies related to this goal, none fitted our scope completely.

Part 3 Overall, 42 invitations were sent to people from 34 different organizations. 22 of them agreed to be interviewed. Finally, 20 interviews with people from 18 different companies were performed, including experts from, e.g., Airbus DS Electronics and Border Security, ETAS GmbH, Hella KGaA Hueck & Co., Robert Bosch GmbH, and ZF TRW Automotive Holdings Corp. The key outcomes are as follows:

Product Engineering Trends: Companies are mainly driven by the increased complexity of system requirements (aspect stated by 60%) as well as by the ever larger number of product variations demanded by their customers (stated by half of the companies).

Importance of Systems Engineering: On a scale from 1 (not important) to 10 (essential for survival), the average importance of Systems Engineering is 7.6 and will increase to 8.5 within the next five years.

Systems Engineering Challenges: 80% stated that change management within the organization is the no. 1 challenge, followed by managing complex requirements and interfaces.

Systems Engineering Process: The larger organizations basically cover every process area of ISO/IEC 15288 and 12207, whereas the SMEs have a clear focus on the technical and implementation processes.

Systems Engineering Practices: Among the already established practices, the companies largely (close to or more than 50%) picked methods, techniques, and approaches related to model-driven development, requirements engineering, test-driven development, and verification and validation.

Specification Languages and Tools: More than 80% of the participants referred to UML as the major relevant specification language. Large organizations tend to use SysML as a more specific language for system modeling. More than 50% of the Systems Engineering tools mentioned were related to modeling different aspects of the overall system or the software as part of the system.

Improvement Potential: The greatest improvement potential for Systems Engineering lies in increased virtual engineering and better integration of the tool chains used, with 50% of the participants mentioning each of these areas.

Systems Engineering Capabilities: The majority of organizations/units rely on internal and external training programs to improve their capabilities related to Systems Engineering. Furthermore, participation in Systems Engineering conferences was mentioned.

Part 4

Based on the key outcomes of the study, a couple of recommendations and areas of activity can be derived for organizations striving towards Systems Engineering:

Organizational Development: Companies should establish a proper change management strategy for introducing Systems Engineering practices and they need to build up appropriate competencies in Systems Engineering in general and Software Engineering in particular. Especially the larger organizations need to think about managing their portfolio of different Systems Engineering projects.

Technical Development: Companies should develop and integrate a Systems Engineering approach including all stakeholders and establish practices in the areas of System Requirements Engineering, Model-Driven Systems Development, and System Verification and Validation. More mature companies should prepare to establish practices in the areas of Virtual Systems Engineering and Integrated Systems Engineering Tool Chains.

Part 5 For all of the five technical development practice areas mentioned above, there already exist established techniques, methods, and tools that cover substantial areas of activity and are applied and have been evaluated in practical settings, or there are techniques, methods, and tools that are currently under development in national and international research and development projects and initiatives.

1 Trends towards Systems Engineering

1.1 Trend towards System Integration

The trend across almost all domains points in the direction of complete integration of systems into so-called Smart Ecosystems, which offer customer-specific solutions across companies driven by a common goal. These Smart Ecosystems break down former insular solutions for the control of business processes and technical processes and make them converge towards an integrated overall solution.

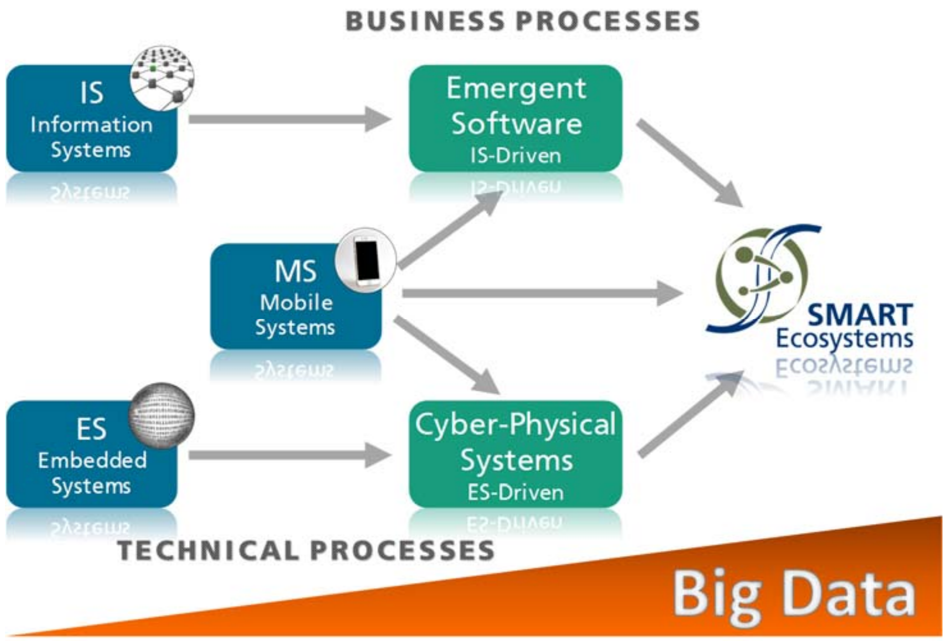

Figure 1: The trend towards Smart Ecosystems

To achieve this, a change of paradigms is going to take place: from monolithic single systems to open, interconnected, scalable, and service-oriented Software Ecosystems. Figure 1 illustrates this trend.

Information Systems (IS) evolve into Emergent Software Systems, which allow for a flexible combination of information systems across system providers and supported business processes. Embedded Systems (ES) evolve into Cyber-Physical Systems (CPS), which allow for a digital representation of physical, real-world objects, making use of dedicated communication infrastructures and the Internet of Things (IoT) to connect systems. In both domains, Mobile Apps are also becoming intensely integrated into business processes today (Naab, Knodel, Kuhn, & Rost, 2016).

A Smart Ecosystem flexibly integrates non-trivial Information Systems used to accomplish business goals with non-trivial Embedded Systems used to achieve technical goals across company boundaries. It functions as one unit to achieve common higher-level goals that no single system would be able to achieve on its own.

The value of data and the potential from using Big Data increases with higher levels of system integration. Good examples of this system integration trend can be found everywhere, such as in the automotive industry (e.g., with Car-2-X communication), in the production area (with Industrie 4.0), in the energy industry (with Smart Energy), in medical technology (with Smart Health), or in agricultural technology (with Smart Farming), and in many other areas. Increasing interconnection is a key factor for innovation and a major contributor to sustainable success (see Figure 2).

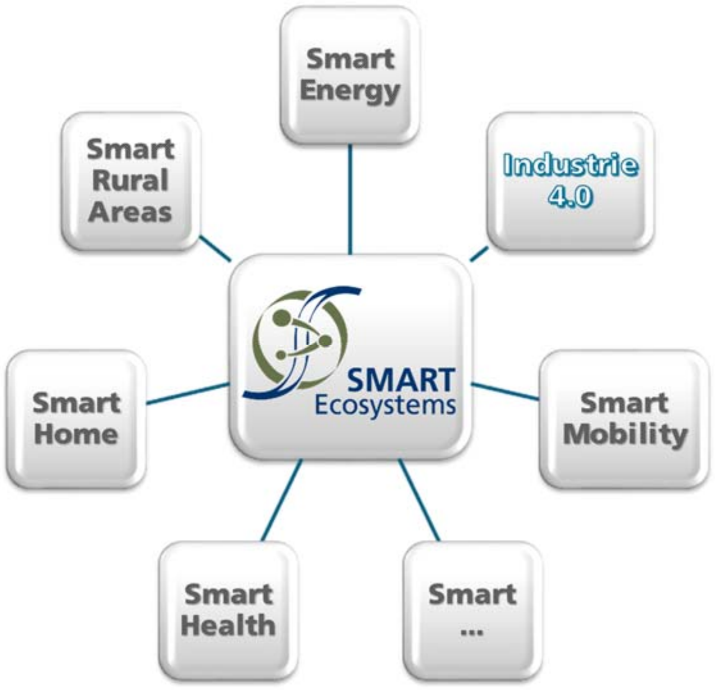

Figure 2: Smart Ecosystem: A trend across domains

1.2 Motivation for Systems Engineering

The basis for the development of such highly integrated systems is a paradigm shift: from monolithic single systems to open, interconnected, scalable, and service-oriented software ecosystems. To allow this vision to become reality, the development organizations behind the systems must change as well. The amount of software in formerly largely hardware-dominated products is continuously increasing. Collaboration across organizational boundaries is a key element for successful software development. The physical world is becoming digital and smart; the Internets of Services, of Things, and of Data are merging with each other.

Software is increasingly used and perceived as an enabler of new, innovative services and business models in all sectors of industry and society. In the future, unique selling points and competitive advantages over competitors will increasingly be generated from interconnecting proprietary products with other systems.

According to the International Council on Systems Engineering (INCOSE), Systems Engineering is an interdisciplinary approach that considers both the business and the technical needs of all customers with the aim of providing a quality product that meets the user's needs.

Being able to establish appropriate Systems Engineering practices in the organization is crucial for staying competitive and for developing innovative products on time, within budget, and with a high level of quality. Specifically, it allows an organization to deal with the typical characteristics of future systems:

(1) Complexity: The complexity of future systems will increase. An organization needs to have means to cope with this complexity. This requires, for instance, model-based engineering approaches instead of textual descriptions, proper systems requirements engineering, scalable architectures that allow for enough flexibility, and mature system development processes.

(2) Diversity: Future systems will most likely comprise and integrate diverse systems and stakeholders across companies and domains. This requires, for instance, interoperable architectures that allow for easy integration, standardization of interfaces, and Quality of Service (QoS) guarantees.

(3) Uncertainty: A system must be able to deal with an uncertain environment: the stakeholders and how to interact with them may change over time. This requires, for instance, highly adaptable systems, the ability to certify certain qualities (such as system performance, functional safety, security, or privacy) at runtime, as well as simulation and virtual engineering approaches in order to connect development time and runtime more closely.

(4) Safety and Security: If highly critical embedded systems are integrated with sensitive information systems, the resulting system needs to address functional safety and security issues at the same time. Otherwise, a security flaw may become a safety issue. This requires, for instance, integrated models addressing security and functional safety at the same time.

(5) User Experience: Despite the rapidly increasing complexity, the systems must stay usable. Product success is more and more dependent on the experience a user makes while using a product. In order to guarantee this user experience, integrated strategies for the user's interaction with the systems are required, for instance.

(6) Autonomy: On the basis of smart data usage, future systems will function increasingly autonomously or semi-autonomously. This requires, for instance, a large degree of (artificial) intelligence and adaptability of the individual systems.

(7) Data-Drivenness: The intelligence of future systems will largely depend on connecting the right data from different sources, analyzing them appropriately, and building models. This requires, for instance, the ability to identify and collect data with an appropriate level of quality on the one hand, and the introduction of powerful data protection mechanisms for guaranteeing an individual's privacy on the other hand.

1.3 Background Information

The following section provides some background explanations about related concepts underlying the trend towards Systems Engineering.

1.3.1 Industrie 4.0

With the advent of industrialization, technology has progressed by leaps and bounds, leading to four industrial revolutions: the first revolution was in the field of mechanization, the second revolution refers to the intensive use of electric energy, and the widespread digitalization marks the third revolution, where every physical "thing" is getting a digital representation. Advanced digitalization, i.e., the combination of the Internet and futuristic technologies in the field of "smart" objects (machines and products) will initiate another paradigm shift, resulting in the fourth industrial revolution a.k.a. Industrie 4.0 (Federal Ministry for Economic Affairs and Energy, 2016), in industrial production.

The Industrie 4.0 strategic initiative was proposed by the German government in the context of the High-Tech Strategy 2020 plan. A variety of terms are used outside German-speaking countries to describe the concept of Industrie 4.0. For example, in the English-speaking world and at EU level, the Internet of Things (IoT) and the trend towards digitalization referred to the third Industrial Revolution. The terms "Smart Production", "Smart Manufacturing", or "Smart Factory" are used in Europe, China, and the US to refer to the digital networking of production to create smart manufacturing systems (Kagermann, Helbig, Hellinger, & Wahlster, 2013).

The Reference Architecture Model for Industrie 4.0 (RAMI 4.0) (Hankel, M.; Bosch Rexroth, 2015) is a unified architecture model that serves the purpose of a common understanding regarding the standards, use cases, etc. that are necessary for Industrie 4.0 and allows discussing associations and details. In RAMI 4.0, Industrie 4.0 components are defined regarding their structure and function. This enables cross-company networking and integration across value-added networks.

This massive integration of data results in technical systems of systems whose capabilities include self-organization, re-organization, and self-optimization. These individualized products constitute a transition from static solutions designed during development time to dynamic solutions that adapt and optimize autonomously during runtime.

1.3.2 Cyber-Physical Systems (CPS)

Cyber-Physical Systems (CPS) are systems evolving from connecting embedded systems with each other and with web-based services (acatech - National Academy of Science and Engineering, 2016). That is, they stand for the connection of the physical and the IT world and result from complex interaction and integration between embedded systems, application systems, and infrastructures, while taking into account Human-Computer Interaction in application processes. They are the technological basis of Industrie 4.0 (Jazdi, 2014).

1.3.3 Internet of Things (IoT)

In a nutshell, "IoT is a novel paradigm relying on the interaction of smart objects (things) with each other and with physical and/or virtual resources through the Internet" (Cavalcante, et al., 2016). The scope of an IoT system varies from a small system with uniquely identifiable "Things" to a system with millions of interconnected "Things" with a physical or virtual representation (e.g., identity, status, location) in the digital world. These "Things" are interconnected using standard protocols, possess sensing/actuation and potential programmability capabilities, and deliver complex services. Taking security into account, the services provided by these "Things" can be made available anytime, anywhere (Minerva, Biru, & Rotondi, 2015).

1.3.4 Cloud Computing

The U.S. National Institute of Standards and Technology (NIST) provides a definition for cloud computing that starts with: "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". The introduced cloud model is composed of five essential properties, three service models, and four deployment models (Mell & Grance, 211).

1.3.5 Internet of Services

IoT and cloud computing converge to provide a set of services called Internet of Services, which represent manufacturing processes. This set of services is published and deployed using cloud-based technologies. Cloud-based manufacturing, for example, is then used to compose services and provide solutions to virtual enterprises (Pisching, Junqueira, dos Santos Filho, & Miyagi, 2015). Customers can request services via a web portal connected to the cloud. After verifying the availability of these services, the background system will then provide these services to the customer to satisfy their needs.

1.3.6 Industrial Internet

Industrial Internet is considered as a sub-paradigm of IoT that focuses more on safety-critical industrial applications (Bruner, 2013). It refers to the integration of complex physical machines with sensors and software in a common network. Technologies that form the basis of the Industrial Internet include "pervasive networks, open source microcontrollers, software that is capable of analyzing massive amount [sic] of data, that understands human preferences and then optimize [sic] across many variables, and the computing power needed to run this intelligence available anywhere at little cost" (Wang, et al., 2015).

1.3.7 Big Data

"With an aggressive push towards "Internet of Things", data has become more accessible and ubiquitous, contributing to the big data environment. This phenomenon necessitates the right approach and tools to convert data into useful, actionable information" (Lee, Lapira, Bagheri, & Kao, 2013). By 2020, it is estimated the digital universe will reach 44 trillion gigabytes of data. The recent trends towards increased digitization and system integration increase the amount of available data called "Big Data" (as depicted in Figure 1), making it a major enabler for new business models and innovation.

Big Data can be characterized according to three dimensions: the *volume* of data, the required *velocity* of providing and processing data, and the increasing *variety* of data. The usage of Big Data supports companies, for instance in making better strategic decisions, controlling processes, understanding customers better, and reducing costs. But it can also be seen as an enabler for the development of new business models (Heidrich, Trendowicz, & Ebert, 2016).

2 Related Studies on Systems Engineering

The analysis of current international studies and publications on usage and best practices of Systems Engineering processes and methods resulted in three surveys and two studies, as described briefly in the following.

2.1 Model-Based Systems Engineering (MBSE) Methodologies

This survey created in 2007 by Jet Propulsion Laboratory and the California Institute of Technology provides a cursory description of some of the leading Model-Based Systems Engineering (MBSE) methodologies used in industry (Estefan, 2007). The intent of the survey was to educate the reader, principally members of the INCOSE MBSE Focus Group, about the various candidate MBSE methodologies that are commercially available and the tools that support the method. The following MBSE methodologies were investigated:

Telelogic Harmony-SE: Harmony-SE: A subset of the larger integrated systems and software development process known as Harmony®. Harmony-SE uses a "service request-driven" modeling approach along with Object Management Group™ Systems Modeling Language™ (OMG SysML™) artifacts.[1]

INCOSE Object-Oriented Systems Engineering Method (OOSEM): OOSEM integrates a top-down, model-based approach that uses OMG SysML™ to support the specification, analysis, design, and verification of systems.

IBM Rational Unified Process for Systems Engineering (RU®P SE) for Model-Driven Systems Development (MDSD): RUP® SE is a derivative of the Rational Unified Process® (RUP®). RUP® is a methodology that is both a process framework and process product and has been used extensively in government and industry. RUP® SE specifically addresses the needs of systems engineering projects. The objective of its creation was to apply the discipline and best practices of the RUP® for software development to the challenges of system specification, analysis, design, and development.

Vitech Model-Based System Engineering (MBSE) Methodology: The Vitech MBSE methodology is based on four primary concurrent SE activities that are linked and maintained through a common System Design Repository. Each of these primary SE activities is linked within the context of associated "domains", where the SE

[1] Remark: At the time of this survey in 2007, Telelogic was an independent company. Meanwhile, Rational has acquired Telelogic and the Harmony Development Process is being further developed as IBM Rational Harmony for Systems Engineering (IBM).

activities are considered elements of a particular kind of domain known as the Process Domain. An MBSE System Definition Language (SDL) is used to manage model artifacts.

State Analysis (SA): State Analysis (SA) is an MBSE methodology that leverages a model- and state-based control architecture, where state is defined to be "a representation of the momentary condition of an evolving system," and models describe how a state evolves. SA provides a process for capturing system and software requirements in the form of explicit models, thereby helping to reduce the gap between the requirements on software specified by systems engineers and the implementation of these requirements by software engineers.

2.2 Improving the Integration of Program Management and Systems Engineering

In this survey, 3,000 INCOSE members (systems engineers) and 5,000 PMI members (program managers) were asked in 2012 about their understanding of how Program Management and Systems Engineering are integrated within their organization and to describe the interactions between the use of standards, integration, formalization, level of effectiveness, and degree of unproductive tension between Program Management and Systems Engineering (Conforto, Rossi, Rebentisch, Oehmen, & Pacenza, 2013).

680 Chief Systems Engineers and Program Managers provided answers. Their organization types were mainly commercial entities (78%), primarily in the US (58%), but also in India, UK, Germany, China, and S. Africa. The industry focus was mainly on professional/scientific (36%), manufacturing (13%), public administration (12%), transportation (6%), and healthcare (4%).

About 30% of the respondents indicated some or significant unproductive tension between Systems Engineering and Program Management. About 20% indicated no unproductive tension. Smaller organizations (below $500 million annual revenue) and large organizations (above $5 billion) are particularly at risk of suffering from unproductive tension. Lack of integrated planning was the key source of unproductive tension. Fully integrated organizations show almost no or only minimal unproductive tension.

The key levers for reducing unproductive tension were: improving the integration of Systems Engineering and Program Management by using standards from both domains, formalizing the definition of integration, developing integrated engineering program assessments, and effectively sharing responsibility for risk management, quality, lifecycle planning, and external suppliers.

2.3 Systems Engineering in Industrial Practice

The objective of this study conducted by Fraunhofer IPT, Heinz Nixdorf Institute, Unity Consulting & Innovation in 2013 was to gain a clear representation of the capability of systems engineering and to obtain the current level of use of SE in practice and in activities in training and further education (Gausemeier, et al., 2015). The current barriers preventing full exploitation of potential benefits are highlighted and recommendations for overcoming them are given. The study is based on 33 interviews with experts from industrial companies and service providers from Germany, Austria, and Switzerland (DACH region). The study participants hold various positions within their companies; primarily CEOs, development managers, production managers, and systems engineers were interviewed. The main results are described below.

The term systems engineering is familiar in practice; most companies have a basic understanding. However, only real experts have a deep understanding. Often, when systems engineering is discussed, the focus is only on software development and is too narrow.

In principle, all participants see considerable potential in the application of systems engineering. Particularly in small and medium-sized companies, the topic of systems engineering has been very person-specific. However, these people, in particular, are mostly keen to transfer the ideas and approaches of systems engineering to their everyday work. Similarly, company-wide awareness of systems engineering is also not yet evident in large companies.

Across all sectors and company sizes, all topics were deemed important, regardless of the systems engineering expertise of the persons interviewed. On average, companies with less systems engineering expertise rated themselves better than companies with more expertise. The following aspects must be overcome here: lack of know-how, lack of a methodical approach, as well as insufficient tool support.

The study proves that, from the industry's perspective, systems engineering is a necessary prerequisite for developing complex technical systems. This concerns not only future systems, which will become increasingly smart and networked, but also current products and product systems to be developed. The multi-disciplinarity of the system, which can no longer be mastered using solely a discipline-specific approach, is an important complexity driver.

In the German-speaking region, the application of systems engineering depends largely on the sector. It has been firmly established in the aerospace industry, as expected, for a long time and is considered indispensable in this area. By now, systems engineering is also regarded as an important "enabler" in the automotive manufacturing industry. German OEMs, in particular, have recognized the potential it offers them to retain their position as system integrators. On the other hand, in the industrial field, and particularly in mechanical and plant engineering, which are largely dominated by small and medium-sized enterprises, systems engineering is largely unknown.

2.4 Systems Engineering Effectiveness

In 2012, the National Defense Industrial Association Systems Engineering Division (NDIA-SED) collaborated with the Institute of Electrical and Electronic Engineers Aerospace and Electronic Systems Society (IEEE-AESS) and the Software Engineering Institute (SEI) of Carnegie Mellon® to obtain quantitative evidence of the benefit of systems engineering (SE) best practices on project performance (Elm & Goldenson, 2012). The objective of the survey was to identify SE best practices used in projects, collect performance data on these projects, and identify relationships between the application of these SE best practices and project performance.

The survey population consisted of projects and programs executed by system developers reached through the NDIA-SED, IEEE-AESS, and INCOSE. About 148 participants completed the survey. The majority of the responses came from U.S. defense industry organizations that were executing contracts within the U.S. for the U.S. Department of Defense (DoD).

Overall, the study found clear and significant relationships between the application of SE best practices to projects and the performance of those projects as shown in Figure 3.

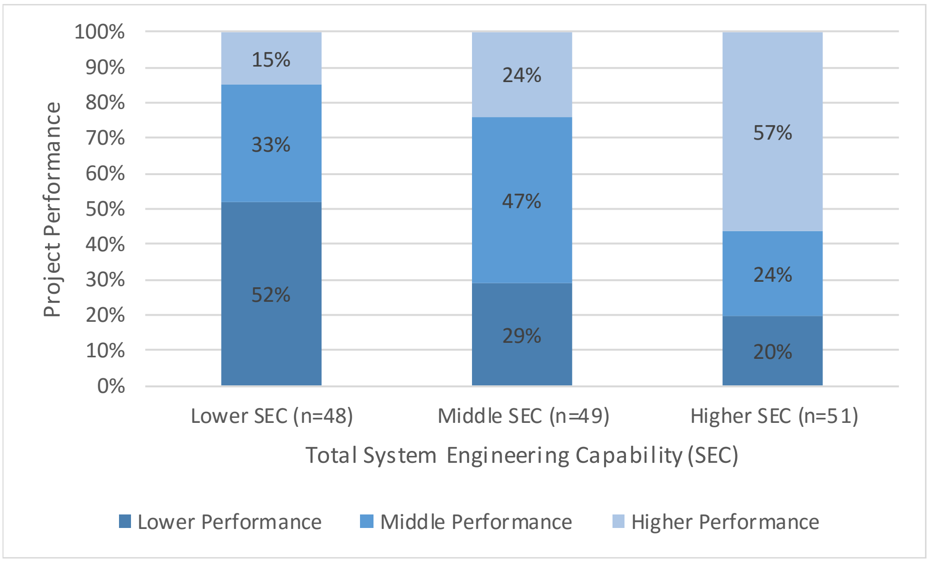

Figure 3: Relationship between Systems Engineering and Performance (Source: Carnegie Mellon University, Software Engineering Institute)

The results of the survey identified those SE process groups that have the strongest relationships to project performance. It also shows that more challenging projects tend to perform worse than less challenging projects. However, projects that face less challenge still tend to benefit from implementing systems engineering best practices. Moreover, the impact of employing systems engineering best practices is even greater for more challenging projects.

2.5 Model-Driven Development

This study was performed in 2013 by Fraunhofer IESE for a large industrial company in Germany in the automotive area that wishes to remain anonymous. The objective of the study was to investigate how to use model-driven development approaches in the practical development of software and systems. Of high interest were the tools used and the integration of tools into a tool chain for implementing a development process. 36 persons working in software development departments in production and research of industrial companies in the automotive domain took part in this study.

Most participants were modeling functional behavior (with Simulink and ASCET) and software structure (with UML/SysML and Simulink/ASCET). Modeling was mainly used for the system architecture and for interfaces. Requirements were usually not modeled very often.

Regarding the usage of models, the study provided the following results: For requirements elicitation, partly Simulink and Visio were being used for structure models (SysML, UML). For functional modeling, Simulink, ASCET, Modelica were being used for executable models, and SysML, UML for structure models. For architectures, Visio was being used for structure models (SysML, UML). For design, UML, Simulink was being used. For testing, Simulink and other tools were being used.

Regarding the creation of models, the study provided the following results: The time investigated in modeling is well invested. Models are often used to validate decisions. Models help to understand algorithms. Models help to detect faults earlier. Automatic generation of code is not always helpful (code quality is often not good enough).

Regarding the quality of the models, many participants stated that SysML/UML models do have faults, inconsistencies, or are not up to date. They also noted that the quality of Simulink models is usually higher than that of SysML/UML models.

3 Systems Engineering Study Results

The goal of the study was on collecting the state of the practice regarding Systems Engineering in the German-speaking area, focusing on challenges and solution approaches in terms of best practices (work processes, methods, and tools). The scope of the study was on Systems Engineering practices across different domains and was not specialized on a single domain. Overall, 42 invitations were sent to people from 34 different organizations. 22 of them agreed to be interviewed. Finally, 20 interviews with people from 18 different companies were performed. 6 organizations/units were classified as a Small or Medium-sized Enterprise (SME) and 14 as a Large Organization (LO). The following companies agreed to be mentioned as a study participant:

Company	**Domains**	**Type**
Airbus DS Electronics and Border Security	Aerospace, electronics	LO
Art of Technology AG	Production, healthcare, aerospace	SME
AVL LIST GmbH	Automotive	LO
Binder Elektronik GmbH	Industry electronics, healthcare	SME
camLine GmbH	Software supplier for production, healthcare, automotive, aerospace, and semiconductors	SME
CIBEK technology + trading GmbH	Solutions for senior citizens, automation technology	SME
ETAS GmbH	Automotive	LO
Hella KGaA Hueck & Co.	Automotive, Electronics, Lighting	LO
Robert Bosch GmbH	Production, automotive, consumer electronics	LO
ZF TRW Automotive Holdings Corp.	Automotive	LO

Table 1: List of study participants

The average duration of an interview was about 60 minutes. A single interview contained 29 questions (12 heading questions with sub-questions). The questions were grouped into four different interview parts dealing with the context of the participant and her/his organization/unit, the challenges related to Systems Engineering they confronted, solution approaches and practices for addressing the challenges, and an outlook to the future improvement potential of Systems Engineering and the organizational capabilities in general.

3.1 Context

Domains: The distribution of domains of the organizations/units can be seen in Figure 4. The majority are from the automotive and production domain, followed by aerospace, transportation, and healthcare. A few companies are from the electronics and mechanical engineering domain.

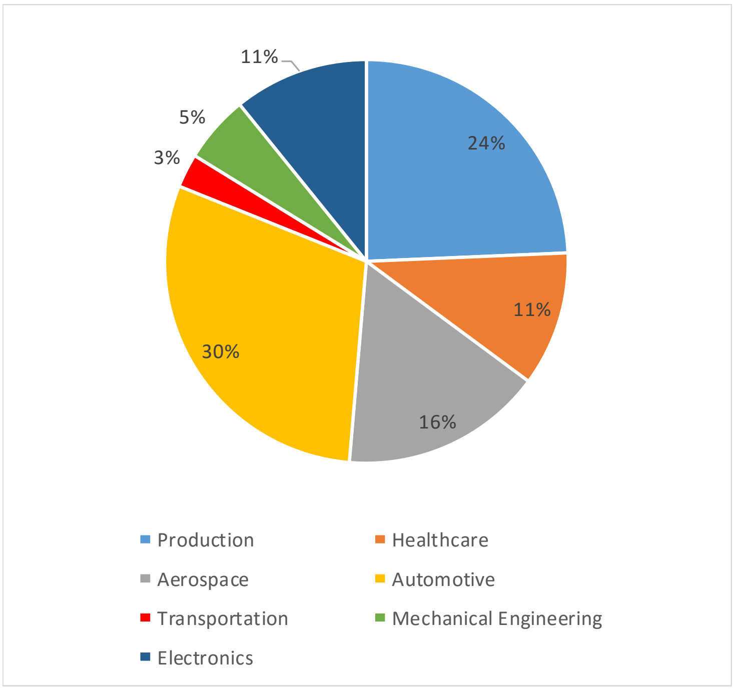

Figure 4: Distribution of participants across domains

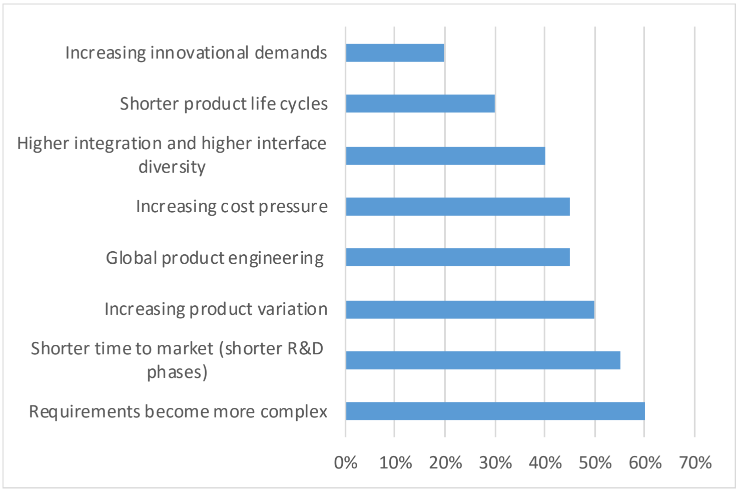

Figure 5: Recent trends

Current trends in product engineering: The companies are currently confronted by many trends in product engineering. Some of them are common, such as increasing requirements complexity (60% of the answers), shorter time to market/shorter R&D phases (55%), and increasing product variation due to customer expectations for individualized products (50% of the respondents). 45% of the respondents mentioned increasing cost pressure and global product engineering as recent trends. A list of the most popular trends is presented in Figure 5.

Future trends in product engineering: Many respondents assume that the current trends will remain relevant over the next five years, but according to the respondents, the leading trends in the future will be the growing multi-disciplinary development, increasing cost pressure, and shorter time to market (each of them was mentioned in 20% of the cases) as can be seen in Figure 6.

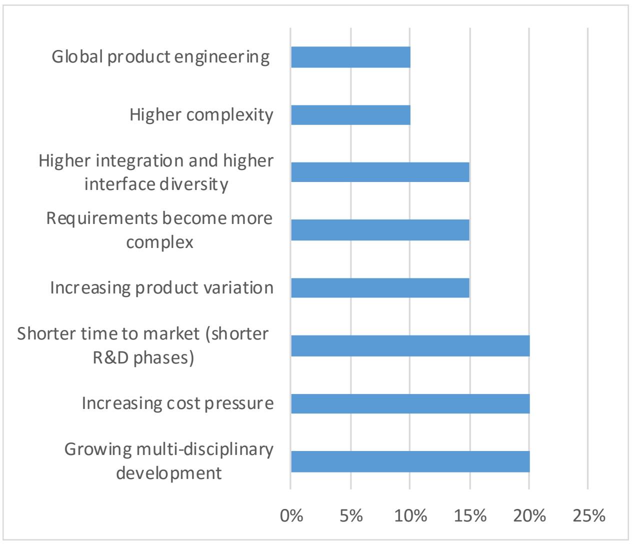

Figure 6: Trends in 5 years

3.2 Challenges

This part talks about practical challenges (e.g., with regard to products, system development processes, organizational structures, required competences) related to Systems Engineering the organization needs to face today and in 5 years from now on.

Importance of Systems Engineering today: On a scale from 1 (not important) to 10 (essential for survival) all organizations state that their implemented Systems Engineering process (including project processes, technical processes, agreement processes, and organizational processes) is important, very important, or essential (cf. Figure 7). The lowest importance value given by the participating organizations is 5 (moderate importance), the highest value is 10 (essential for survival). The average importance is 7.6, meaning important.

For about 25% of the participating organizations, this process is really essential (importance value 9 or 10). About 35% of all participating organizations stated that Systems Engineering is only of moderate importance (importance value 5 or 6). All other participating organizations gave values in between. The standard deviation is 1.5, meaning that the answers given are within a narrow margin.

There is no organization where Systems Engineering does not play any role or just a minor role. No significant difference between large organizations and SMEs can be discovered.

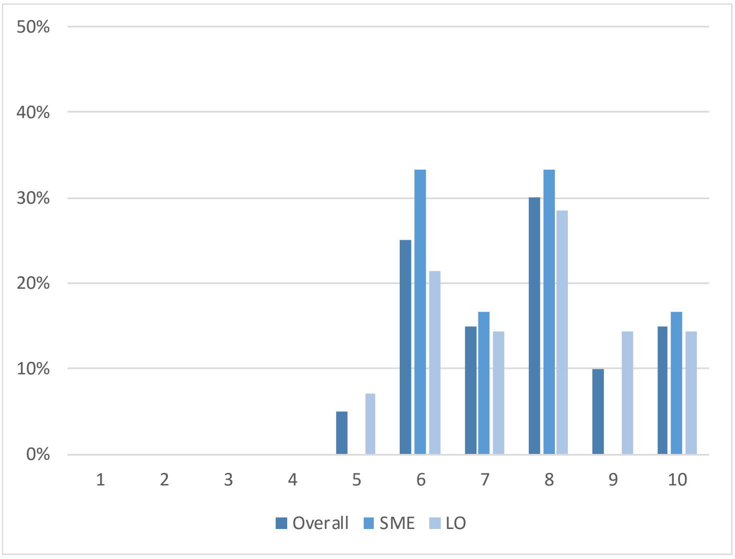

Figure 7: The importance of Systems Engineering today

Importance of Systems Engineering in 5 years: Nearly all organizations estimate that Systems Engineering will become more important in the future. The average importance is increasing significantly from 7.6 to 8.7 within the next 5 years (cf. the upwards shift of the importance values from Figure 10 to Figure 11). This increase is seen in general across all types and sizes of organizations and also across all application domains. Nevertheless, it can be seen in Figure 11 that large organizations generally estimate a higher importance value in 5 years compared to SMEs. This can be interpreted such that Systems Engineering will play a more important role in large organizations in the future than in SMEs.

The standard deviation decreases from 1.5 to 1.1, which means that the estimated importance in 5 years is even more focused around the average importance of 8.7 (more organizations estimate the same higher importance).

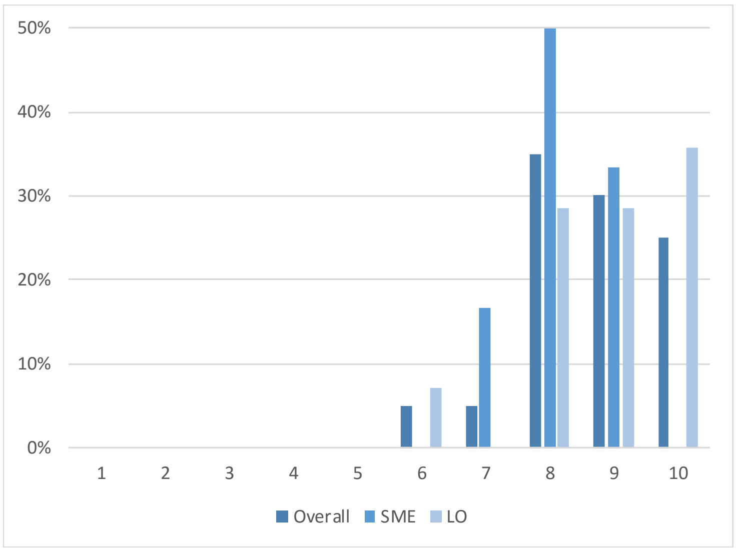

Figure 11: The importance of Systems Engineering in 5 years

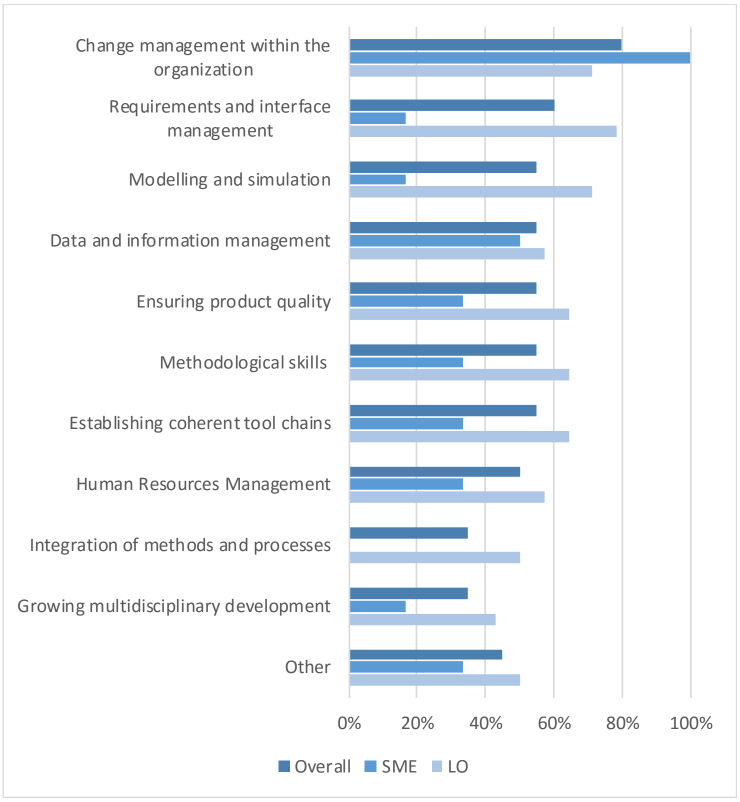

Figure 8: Current Systems Engineering challenges

Current Challenges in Systems Engineering: Change management within the organization is the top challenge that nearly all organizations are currently confronted with, followed by requirements and interface management (cf. Figure 8). Other important challenges are modeling and simulation, data and information management, ensuring product quality, establishing/keeping methodological skills within specialist disciplines and across disciplines, establishing coherent tool chains, and human resources management.

SMEs are mainly confronted with change management within the organization, methodological skills within specialist disciplines and across disciplines, establishing coherent tool chains within the organization and across organizational boundaries, and ensuring product quality (e.g., reliability, safety, security).

Large organizations are mainly confronted with requirements and interface management of complex systems or even systems of systems, modeling and simulation, change management within the organization creating acceptance of new approaches and technologies, establishing coherent tool chains within the organization and across organizational boundaries, and ensuring product quality (e.g., reliability, safety, security).

Additionally, nearly every organization is fighting other individual challenges depending on its current product roadmap, process organization, infrastructure, or tool chain. In Figure 8, these individual challenges are summarized in the category "Other".

Future Challenges in Systems Engineering: For most organizations, challenges related to Systems Engineering within the next five years are largely the same challenges they are confronted with today. Additional future Systems Engineering challenges are highly diversified, depending on the application domain and the individual system development processes of each organization.

On the technical process level, these challenges range from model-based development via agile development or rapid prototyping to verification and validation with virtual prototyping and simulations. Better requirements and interface management for upcoming systems of systems was also mentioned.

On the project process level, new challenges such as introducing more product variants or keeping up the product quality (especially w.r.t. security and safety) are becoming more important. Here, SMEs have a special interest in data and information management and in introducing change management in the organization.

On the organizational process level, new challenges such as improved change management aimed at handling the transformation process of digitalization in the company or close leadership to really perform the Systems Engineering processes seem to be important. Some organizations plan to place more emphasis on human resources management in order to build up and preserve Systems Engineering know-how.

No specific trend can be discovered here across the type or size of organization. Nevertheless, specific trends towards new functionality exist in individual application domains; e.g. in the automotive domain, autonomous driving creates new challenges for requirements and interface management for networked systems, modeling and simulations of products/solutions, and safety requirements.

3.3 Solution Approaches

After discussing the major challenges related to System Engineering this part discusses the most promising solution approaches taken by industry. This comprises the use of best practices, standards, methods, tools, etc.

System Engineering Process: The larger organizations basically cover every process area of ISO/IEC 15288 and 12207, whereas the SMEs have a clear focus on the technical and implementation processes. The standards they adhere to are quite domain-specific, except for quite general approaches such as ISO 9001. 40% of the larger organizations explicitly referred to ISO/IEC 15288.

Regarding the process model used, more than 45% of the large organizations and SMEs claim that they are following an agile model, whereas more than 50% of the large organizations follow a waterfall model or iterative waterfall model. Furthermore, more than 80% of the large organizations provide different variants of their standard process.

The majority of the SMEs (83%) have defined a common development process with little variants. In 86% of the larger organizations, several variants of the development process exist. However, in general a standard development process is defined that is tailored according to project needs.

Stakeholder Involvement: In large organizations, the different stakeholders and disciplines are interlinked and coordinated by following a defined process (85%). Personal communication is the preferred way of smaller companies to organize their product development (20%).

Workshops and the creation of mixed teams to get a common project understanding are established in all organizations (SMEs: 80%, LOs: 57% resp. 50%). The use of tools and common data pools across organizational boundaries is an issue for all organizations (SMEs: 20%, LOs: 50%).

The integration of external suppliers into development activities is mostly performed by supplier agreements (SMEs: 40%, LOs: 93%). A closer relationship is established by larger organizations through subcontractor management (43%).

In addition, body leasing concepts are applied by LOs on a large scale (64%) to provide external knowledge for development activities. Training activities including workshops together with external suppliers are the means by which smaller organizations get a common understanding in development activities (40%).

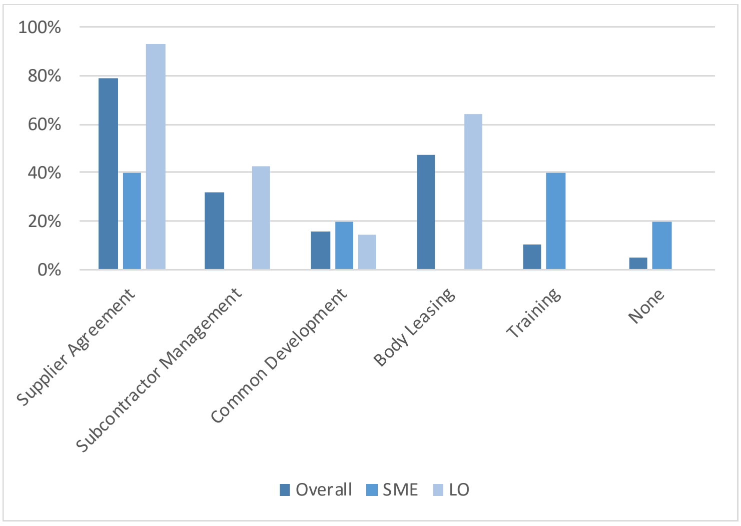

Figure 9: External supplier integration

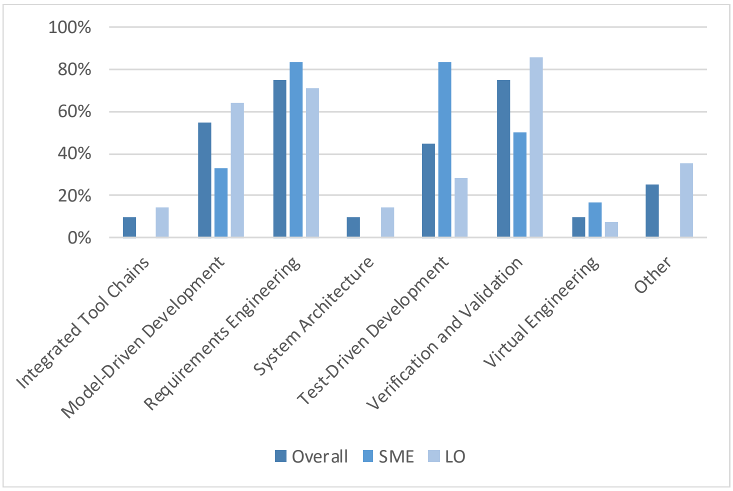

Figure 10: Top Established Systems Engineering Practices

Top 3 Established Practices: As can be seen from Figure 10, from the already established practices, the companies largely (close to or more than 50%) picked methods, techniques, and approaches related to model-driven development, requirements engineering, test-driven development, and verification and validation. Further practices mentioned by at least more than one organization include integrated tool chains and virtual engineering, and an overall system architecture.

Moreover, the selection of the top practices varies between large organizations and SMEs. The variance is especially large for model-driven development and for system verification and validation, which were chosen by more than 60% and 80% of the large organizations, respectively, but only by less than 40% and about 50% of SMEs, respectively. More than 80% of the SMEs picked test-driven development as a top established practice. This was only picked by about 30% of the large organizations. Please note that this does not mean that large organizations do not do test-driven development; it only means that this was not picked as one of the top three practices.

Regarding the most strongly impacted process areas, the participants agreed with a huge majority that the technical and software implementation process areas (as defined by ISO/IEC 15288 and 12207) are impacted by the practices.

Languages for System Modeling: As can be seen in Figure 11, the majority referred to UML as the major relevant modeling language. Large organizations tend to use SysML (based on UML) as a more specific language for system modeling. Furthermore, some domain-specific languages were mentioned. Singular answers included DFD (Data Flow Diagrams), FMI (Functional Mock-up Interfaces), OSLC (Open Services for Lifecycle Collaboration), Structured Analysis, XML/XMI, IDef0, and Autosar.

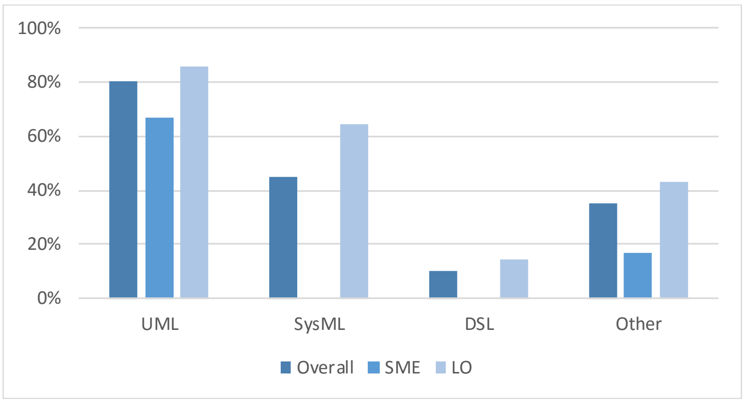

Figure 11: Used Specification Languages

Systems Engineering Tools: Overall, more than 90 statements about tool usages related to Systems Engineering were made by the study participants (80% of them stem from large organizations) and over 40 different tools or components of tools were among these statements.

The vast majority of tools mentioned is related to modeling different aspects of the overall system or the software as part of the system. Depending on the domain, some tools were quite domain-specific (such as appropriate CAD software). Furthermore, mostly requirements-specific simulation tools and testing tools were mentioned to support the previously listed practices.

Regarding modeling tools, about 50% of the participants stated that they are using "Enterprise Architect" and "MATLAB". Regarding requirements tools, 30% use "DOORS" and "Microsoft Office". Regarding simulation tools, 40% use MATLAB's "Simulink" extension. Regarding testing tools, a variety of different tools were mentioned.

Close to 90% of the tools or components mentioned were specific for a certain type of activity, whereas a bit more than 10% were multi-purpose tools or integrated tool suites. Furthermore, close to 10% of the answers mentioned self-developed tools. Mostly this was used in the area of simulation (about 5%).

Emerging Technologies and Needs: The most prominent emerging technologies, with close to 40%, were the adoption of more formal methods and model-based system development approaches instead of informal/textual specifications. Furthermore, the general need for better integration of tool chains, virtual engineering incl. simulation, and the development of their own specialized tools were mentioned as technological areas for the near future. Moreover, some general more product-/feature-related trends were mentioned that have an impact on the choice of technologies, such as Big Data, Internet of Things, and service orientation.

3.4 Outlook and Capabilities

Improvement potential: As can be seen in Figure 12, the greatest improvement potentials for Systems Engineering, with more than 50% each, are in increased virtual engineering and better integration of the tool chains used. The demand seems to be bigger for SMEs.

For nearly 40% of the larger organizations, improved program management (aka. project portfolio management) is also worth mentioning. This is no surprise as larger organizations need to deal with a larger number of projects running simultaneously. For close to 40% of the SMEs, a higher degree of automation was seen as an important improvement potential.

Improving System Engineering Capabilities: Regarding ways to improve an organization's own capabilities in Systems Engineering, a variety of answers were given. However, as can be seen in Figure 13, the vast majority relies on making use of internal and external training programs. Not surprisingly, nearly all of the organizations offer such training programs internally. Furthermore, participation in Systems Engineering conferences for the purpose of exchanging knowledge and experience among peers and with researchers and discussions about trends and solution approaches was mentioned by more than 50% of the overall participants and by more than 60% of those from larger organizations.

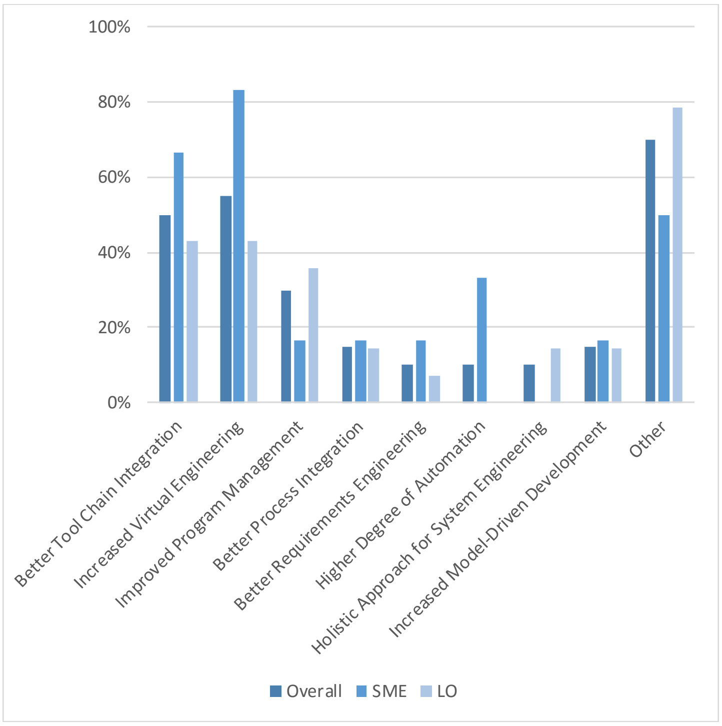

Figure 12: Areas of Improvement Potential for Systems Engineering

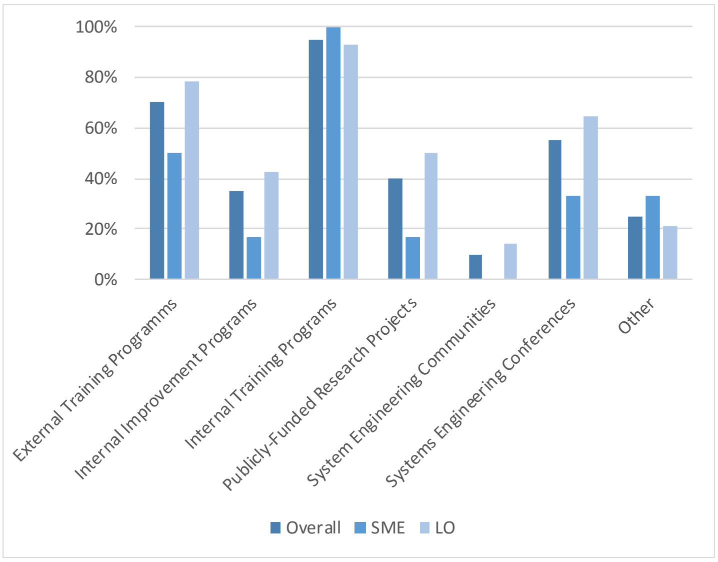

Figure 13: Approaches for Improving System Engineering Capabilities

3.5 Discussion of Potential Threats and Limitations

This part discusses a summary of the major potential threats and limitations of the performed study related to the methodology applied and their performers:

(1) Sample size: The overall number of interviews performed is fairly small. This limits the generalizability of the results on the one hand, but also our possibilities of analyzing relationships among the answers on the other hand. However, because of the relatively small amount of interviews, it was possible to focus more on the single answers provided and to get deeper insights into single cases.

(2) Self-reported data: The participants of the study were asked to report about their specific knowledge and about experience limited to their specific context in the organization/unit. They were asked to explicitly answer based on their first-hand experiences and not to make assumptions about what is going on outside their responsibilities and fields of expertise. However, the answers given are still biased by their personal perception. As only one person of an organization or of a specific unit of a (larger) organization was interviewed, there was no chance to analyze discrepancies among answers; the researcher had to trust what was said about the organization/unit.

(3) Questionnaire: The questionnaire used to guide the interviews was systematically derived from the goals of the study and peer-reviewed internally by Fraunhofer IESE researchers. During the first two interviews, it was experienced that some questions were hard to answer without some further hints about the intention of the question. Furthermore, the interviews took longer than initially planned (60 minutes at most). For that reason, example answers (mostly containing potential alternatives) for 13 out of the 29 questions were provided. The example answers were created based on the related studies previously analyzed and on the experience of the Fraunhofer IESE researchers from past Systems Engineering projects. However, it was made clear that the interviewee should not just select from the provided examples, but should also be encouraged to think beyond them. The consequence was that coding of the provided answers was simplified as many answers could be mapped to the existing list. This contributed to facilitate comparability of the answers. Furthermore, the time for conducting an interview could be significantly reduced.

(4) Trust and openness: The Fraunhofer-Gesellschaft is well known in Germany as an objective, neutral, and independent partner. This guarantees a certain openness towards participating in a study as well as openly talking about challenges and solution approaches (at least if it is not conflicting with the core intellectual property of the company). Furthermore, it was made clear in the invitation to the interview for what purpose the results would be used, that the minutes would be anonymized before being analyzed, and that the interviewees

would have the chance to review the minutes and would have to explicitly approve the use of the minutes as part of the analysis.

(5) Language: All interviews were performed in German and then translated into English for further analysis. After translation the interviews were sent to the participants for approval. This gave the interviewees the chance to check the accuracy of the translation to their best knowledge and was done to confirm that the interview minutes reflect the opinion of the interviewees properly. Furthermore, the interviewees had the chance to make extensions and corrections to the given answers.

4 Study Key Outcomes and Recommendations

4.1 Key Outcomes

The following key outcomes can be extracted from the 20 interviews based on our analysis:

(1) Product Engineering Trends: Companies are mainly driven by the increased complexity of system requirements (aspect stated by 60%) as well as by the ever larger number of product variations demanded by their customers (stated by half of the companies). In combination with shorter time to market (about 55%), this puts a lot of pressure on current system engineering. In the future, more cross-disciplined development is seen (by 20%) as an additional driving factor, which will in turn increase the complexity of projects. The trend to increasing cost pressure and shorter time to market is expected to remain.

(2) Importance of Software: More than 85% of the companies stated that software plays a major role in their products; even though about 70% of the participants stated that they come from a pure hardware development world. Furthermore, 85% stated that they spent 30% or more (up to 90%) of the development budget on software development. More than half of the participants agreed that this will further increase within the next five years.

(3) Importance of Systems Engineering: On a scale from 1 (not important) to 10 (essential for survival), the average importance of Systems Engineering is 7.6. Though Systems Engineering is currently already very important, this will increase to 8.7 within the next five years. Most participants stated that the reason for the increasing importance are customer demand for higher product quality in combination with increased complexity of the products. This especially refers to requirements related to system platforms and system integration.

(4) Systems Engineering Challenges: 80% stated that change management within the organization is the no. 1 challenge, followed by managing complex requirements and interfaces (especially for systems of systems). Additional future Systems Engineering challenges are human resources management, the transformation and organization processes regarding Systems Engineering within the organization and data- and information management.

(5) Systems Engineering Process: The larger organizations basically cover every process area of ISO/IEC 15288 and 12207, whereas the SMEs have a clear focus on the technical and implementation processes. The standards they adhere to are quite domain-specific, except for quite general approaches such as ISO 9001. 40% of the larger organizations explicitly referred to ISO/IEC 15288. Regarding the process models used, more than 45% of the large organizations and SMEs claim that they are following an agile model, whereas more than 50% of the large organizations follow a waterfall model or iterative waterfall model. Furthermore, more than 80% of the large organizations provide different variants of their standard process.

(6) Multiple Stakeholders: There are many different disciplines and corresponding stakeholders involved in the Systems Engineering process across all organizations regardless of their size. However, classic engineering disciplines like Hardware Engineer or Software Engineer are still viewed as "isolated" disciplines within the organizations. The particular role of "Systems Engineer" is only defined in larger organizations. In 85% of the large organizations, different stakeholders and disciplines are interlinked and coordinated by following a defined process. For SMEs, this figure is less than 20%. Instead, personal communication is the preferred way of smaller companies. Between 60% and 70% of the companies create joint teams and perform joint workshops and meetings for coordination purposes.

(7) External Suppliers: Almost 60% of the organizations get less than 25% of their product parts supplied from external sources. Nevertheless, one third of the organizations obtain up to 50% from external suppliers. The average proportion of externally supplied product parts is about 25% across all organizations. The average criticality in terms of intellectual property of externally supplied components is 3,5 on a scale from 1 (not critical) to 10 (highly critical).

(8) Systems Engineering Practices: Among the already established practices, the companies largely (close to or more than 50%) picked methods, techniques, and approaches related to model-driven development, requirements engineering, test-driven development, and verification and validation. Further practices mentioned by at least more than one organization include integrated tool chains, virtual engineering, and an overall system architecture. Whereas large organizations focus on model-driven development as well as system verification and validation, which was chosen by 60% and 80%, respectively, around 80% of the SMEs picked test-driven development as their top established practice.

(9) Impacted Processes: The participants agreed with a huge majority that the technical and software implementation engineering process areas (as defined by ISO/IEC 15288 and 12207) are mostly impacted by Systems Engineering practices.

(10) Specification Languages and Tools: More than 80% of the participants referred to UML as the major relevant specification language. Large organizations tend to use SysML as a more specific language for system modeling. Furthermore, domain-specific languages were mentioned in general. More than 50% of the Systems Engineering tools mentioned were related to modeling different aspects of the overall system or the software as part of the system. Furthermore, 30% mentioned requirements and 40% simulation tools as being relevant. Moreover, close to 10% of the answers mentioned self-developed tools. Mostly these are used in the area of simulation (about 5%). Moreover, close to 40% mentioned the adoption of more formal methods and model-based system development approaches instead of informal/textual specifications as a technological area to be addressed in the near future.

(11) Improvement Potential: The greatest improvement potential for Systems Engineering lies in increased virtual engineering and better integration of the tool chains used, with 50% of the participants mentioning each of these areas. The demand seems to be bigger for SMEs. For nearly 40% of the larger organizations, improved program management (aka. project portfolio management) is also worth mentioning. For close to 40% of the SMEs, a higher degree of automation was seen as an important improvement potential.

(12) Systems Engineering Capabilities: The majority of organizations/units rely on internal and external training programs (close to 100% and more than 60%, respectively) to improve the capabilities related to Systems Engineering. Furthermore, participation in Systems Engineering conferences was mentioned by more than 50% of the overall participants and more than 60% of those from the larger organizations.

4.2 Recommendations and Areas of Activity

From the given 12 key outcomes of the study, a few recommendations and areas of activity can be derived for organizations striving towards Systems Engineering. Please note that these recommendations and actions are motivated by the study outcomes, but they are somewhat subjective as there may be other strategies for reaching the same goal.

We split the recommendations and areas of activity into those more closely related to organizational development and those more technically related to how organizations develop systems.

4.2.1 Organizational Development

(O1) Change Management Strategy: 80% of the companies stated that change management within the organization is the key challenge for Systems Engineering (see outcome #4). Therefore, it is important to openly think about which organizational structure and processes are best suited for coping with Systems Engineering challenges. In particular, it is important to include all stakeholders in that process in order to gain acceptance and to better motivate/communicate changes and carefully plan how these changes should happen (see outcome #6).

(O2) Systems Engineering Competencies: Creating internal and buying-in external training programs on different Systems Engineering topics was obligatory for the majority of organizations (see outcome #12). Additionally, we would recommend that organizations participate in Systems Engineering conferences and become active members of corresponding communities in order to get information about recent developments and exchange experiences regarding Do's and Don'ts (see outcome #12).

(O3) Software Engineering Competencies: As 85% of the companies stated that software plays a major role in their products although they come from a more hardware-oriented development world (see outcome #2) and as this will increase in the future, it is important for companies to build up or maintain an appropriate number of Software Engineering competencies. This number depends on the degree to which their product depends on software and what the major IP (intellectual property) and USP (unique selling point) of the company is. If the IP/USP is in software or is becoming software, it would make sense to build up their own resources in the area of Software Engineering. If software is only a means to an end, it makes at least sense to build up competencies for managing external software suppliers and partners (see outcome #7).

(O4) Project Portfolio Management: Larger organizations should place special focus on the management of the overall portfolio of their projects and the interconnections and dependencies among them, as this was mentioned as a special issue for improvement (see outcome #11).

4.2.2 Technical Development

(T1) Integrated Systems Engineering Approach: As time to market for new products is getting shorter and product complexity is increasing at the same time (see outcome #1), it is important to efficiently and effectively deliver value to the customers. Systems Engineering is considered very important for dealing with this issue, especially when it comes to system platforms and system integration (see outcome #3). This requires a well-integrated and aligned approach across all disciplines involved (see outcome #6). Especially when it comes to technical and implementation processes, companies should carefully think about what impact Systems Engineering has (see outcome #9) and – as there are no silver bullet approaches – what a custom-tailored process should look like that best fits the needs of the individual organization (see outcome #5).

(T2) System Requirements Engineering: The complexity of system requirements and the number of product variants has increased over time. As a matter of fact, in the near future they will further increase as (even) more cross-disciplined development will come into play (see outcome #1). This forces companies to think about how to elicit/develop requirements on the system level and how to manage them systematically over time. This also includes how to break them down into lower-level (especially software) requirements (see outcome #2).

(T3) Model-driven Systems Development: The study confirmed that model-driven development of systems is seen as a key practice for an organization. Larger organizations have already implemented it at least partially (see outcome #8) or see this as an essential improvement potential (see outcome #11). The actual use of formal modeling languages varies, even though there are very prominent ones such as UML and SysML. In the area of tool support, a variety of tools were mentioned as well (see outcome #9). An organization should therefore carefully evaluate which aspects of the system specification to model and what appropriate language and tool support is available. This tool selection should also be influenced by the interfaces provided by suitable tools to ensure seamless integration into the tool landscape of the development process (see outcome #10 and T6).

(T4) System Verification and Validation: Companies should think about establishing proper techniques and methods for system verification and validation and specifically for test-driven system development, as these areas were seen as crucial by many organizations (see outcome #8). Additionally, the development process should ensure that system verification and validation is properly linked to system requirements at all times.

(T5) Virtual Systems Engineering: As the complexity of products is increasing (see outcome #1) and development is becoming more multi-disciplined (see outcome #6), it becomes difficult and very cost-intensive to compose the different system parts physically. Therefore, companies should think about the feasibility of using virtual engineering systems based on sound models. In the future, this is seen as a major improvement potential for speeding up development (see outcome #11). Some companies have already introduced or developed their own simulation tools for system verification and validation (see outcome #10).

(T6) Integrated Systems Engineering Tool Chains: As we have observed, a variety of different tools are used for Systems Engineering in the organizations. Furthermore, companies have developed their own tools for particular tasks and for overcoming the shortages of existing tools (see outcome #10). One major point for improvement is better integration of the tool chains (see outcome #11)., Especially when starting to do Systems Engineering, companies should therefore put special emphasis on the interoperability of their tools and on having as much integration across the tool chain as possible.

5 Selected Industrial Practices and Cases

Based on the results of the study, this part will give more details on how selected best practices for Systems Engineering can be implemented in the organization. Among the already established practices, the companies largely (close to or more than 50%) picked methods, techniques, and approaches related to the following areas:

1. Model-driven System Development
2. System Requirements Engineering
3. System Verification and Validation

As the greatest potential improvement areas for Systems Engineering, the following areas were mentioned by 50% of the companies:

4. Systems Engineering Tool Chain Integration
5. Virtual Engineering of Systems

Whereas we have some evidence that the former three are at least partially applied at more than half of the interviewed companies, the latter two still seem to be at the beginning of their practical application and implementation, but are considered to have great potential for Systems Engineering. Therefore, in the following, for each of these five practice areas, we will give a brief description, highlight some concrete examples of methods, techniques, and approaches, present industrial cases of their implementation and application in real settings, and summarize some recommendations and lessons learned.

5.1 Model-driven System Development

The development of modern systems is becoming increasingly difficult and challenging. Domains such as automotive and avionics demand high integrity levels between hardware and software to ensure proper execution of their systems. A commercial airplane, for example, contains systems that control ground proximity, navigation, and engine commands, amongst others. Because of that, it is important to ensure that each aspect of the system is properly described and understood.

Model-driven System Development (a.k.a. Model-based Systems Engineering) is a system development approach that is based on the refinement of models. This refinement of models usually happens on different abstraction levels until such a level of detail is achieved that the system can be implemented immediately, or, ideally, extracted automatically from the models. It usually provides a set of integrated modeling techniques and tools to support all substantial development disciplines, starting with model-based requirements engineering, via model-based design and model-driven implementation (which can be partially automated by extracting code from models) to the certification of these systems (using modeling techniques such as fault trees).

Model-driven Software Development is a part of the overall model-driven system development that incorporates different techniques across the entire spectrum of software development activities, including model-driven requirements engineering, model-driven design, code generation from models, model-driven testing, model-driven software evolution, and more.

Model-driven system development is an important technique for managing the complexity of modern systems. It provides a set of integrated modeling techniques and tools to support all substantial development disciplines including the certification of these systems (e.g., using modeling techniques such as fault trees).

5.1.1 Example Approaches

The SPES 2020 methodology (Pohl, Achatz, & Broy, 2012) for the development of embedded systems is a concrete method for model-driven system development. The SPES (Software Platform Embedded Systems) 2020 initiative was a joint research and development project between academia and industry partners from different domains like avionics, automotive, health care, and energy. The SPES 2020 modeling framework organizes the development artifacts of model-driven system development into four architecture viewpoints.

Requirements Viewpoint: This view aims at supporting the requirements engineering process in eliciting, documenting, and managing the system requirements. In SPES, the elicitation of requirements starts with the identification of the system context, such as users, stakeholders, and external systems that somehow interact with the system. These entities are documented in a system/context diagram of the requirements viewpoint, which shows the system as a black box and documents the interaction of the system with its environment. This diagram complements traditional RE techniques (like scenarios) and helps in eliciting functional and quality requirements, business drivers, and constraints (e.g., legal constraints), which are all documented in the requirements viewpoint.

Functional Viewpoint: Every system has a set of functions that each offers a particular service to the users of the system. One of the main purposes of the functional viewpoint is to identify and formalize these functions. In the SPES methodology, the context diagram and the scenarios from the requirements viewpoint are used as the starting point for identifying the user functions. The functional viewpoint then formalizes these user functions by defining the input/output behavior, e.g., by means of functional dataflow diagrams. The functional viewpoint is also the place where dependencies between functions are identified and where a refinement of user functions into sub-functions takes place.

Logical Viewpoint: Once the functional model has been established, the next step is to identify which of the functionalities are to be implemented by software or hardware (or as a mixture). Hence, the logical model describes how the functionality of the system (as identified in the functional perspective) should be decomposed into a network of communicating and cooperating components. The logical viewpoint is the first place where design decisions should be taken and is hence solution-oriented, whereas the functional viewpoint is ideally only problem-oriented.

Technical Viewpoint: In this viewpoint, the hardware and software elements are detailed in implementation entities that realize the logical components. Besides the detailed software design, this view includes the Hardware Network View, which describes networks of hardware elements such as buses, sensors, and actuators, and the Deployment View, which shows the deployment strategy of logical software components to hardware entities.

While the views sketched above seem to indicate a waterfall-like process from the requirements viewpoint to the technical viewpoint, the SPES methodology is actually more iterative. Usually, one starts by eliciting requirements on a high abstraction level, which are formalized in the functional model and realized at the logical/technical level. Each of the identified components at the logical level can itself be regarded as a system under discussion with its own more concrete requirements, functions, logical and technical solutions.

5.1.2 Practical Cases

The methodology was successfully applied at industry companies from different domains (Pohl, Achatz, & Broy, 2012). We describe a practical case in the context of developing the control software of a large telescope array that monitors gamma rays in the universe (Achary & Actis, 2013). The architecture model was created using a refined UML profile that supports the SPES viewpoints (Kuhn & Antonino, 2014). A detailed description of the work can be found in (Oya, et al., 2016).

The Cherenkov Telescope Array (CTA) is an initiative to build two large arrays of Cherenkov gamma-ray telescopes. It will serve as an open observatory to a wide astrophysics community and will provide a deep insight into the non-thermal high-energy universe. Cherenkov telescope systems use the effect that gamma rays produce particle cascades that emit so-called Cherenkov light showers, which can then be detected by cameras hosted by ground-based telescopes. The aims of CTA can be roughly grouped into three main themes serving as key science drivers: understanding the origin of cosmic rays and their role in the universe, understanding the nature and variety of particle acceleration around black holes, and searching for the ultimate nature of matter and physics beyond the standard model.

The array control and data acquisition (ACTL) project within CTA will deliver the software to control and acquire the data from the CTA instrumentation. The objective is to create and maintain a single architecture model that will allow tight integration of software development and coordination processes and decisions. First, the model will provide the main input for managing the project management organization, for example for generating a work breakdown structure (WBS), creating the effort estimates, evaluating the risks, and providing input for the definition of priorities and the identification of unplanned work. It will also provide specifications/contracts for the developers of the team so they can understand the context of the software to be developed, and will hence be used to automatically generate the developer documentation. Furthermore, the model should drive the verification and validation process to test the code provided by the developers. Finally, it will allow communicating the requirements, decisions, and adopted technical solutions inside and outside the work packages.

5.1.3 Lessons Learned and Recommendations

Based on (Pohl, Achatz, & Broy, 2012), a few recommendations and lessons learned can be derived from the practical evaluation in the different domains:

- The SPES methodology provided traceable and seamless support for all engineering life cycle phases with various levels of integration depending on the domain. The highest level of integration was achieved in the automotive area.
- It allowed for early consideration and verification of system properties, addressing users' expectations regarding completeness, consistency, safety, or traceability.
- The methodology addressed safety, standard compliance, and certifiability needs. For instance, in the automotive domain, the logical architecture allowed for automated transformation into AUTOSAR application components. The integrated design and safety modeling showed that engineers can seamlessly work on the same model.

Furthermore, it was concluded that integrated development such as supported by the SPES methodology is essential for the engineering of embedded systems.

5.2 System Requirements Engineering

Several institutions provide definitions of the term Requirements Engineering (RE). The International Requirements Engineering Board (IREB) defines it as: "Requirements engineering is the systematic and methodologically sound approach to requirements analysis and management" (Sophist). The IEEE defines RE as: "Requirements Engineering is the branch of systems engineering concerned with managing desired properties and constraints of software-intensive systems and with goals to be achieved in the environment. It is concerned with these aspects from the problem analysis stage to the implementation and maintenance stages of a system. Additional variety is added because of differences in issues that arise in different domains, ranging from public administration software to workflow systems, groupware and embedded systems and control software".

The goal of RE is to develop good requirements and to manage them during development with respect to risks and quality. RE is the discipline within systems and software engineering that bridges the entire life cycle and thus determines the success or failure of a product or project. It is an engineering discipline because of its disciplined and systematic approach (Ebert, 2014).

5.2.1 Example Approach

The processes used for RE vary widely depending on the application domain, the people involved, and the organization developing the requirements. The following generic activities are common to all processes according to (Software Engineering Institute, Carnegie Mellon):

- Requirements Elicitation: The process of discovering, reviewing, documenting, and understanding the user's needs and constraints for a system.
- Requirements Analysis: The process of refining the user's needs and constraints.
- Requirements Validation: The process of ensuring that the system requirements are complete, correct, consistent, and clear.
- Requirements Specification: The process of documenting the user's needs and constraints clearly and precisely.
- Requirements Change Management: The process of scheduling, coordinating, and documenting the requirements engineering activities (that is, elicitation, analysis, specification, and verification).

For each of these generic activities, lots of techniques exist that can be applied to execute the processes. A good overview of best practices can be found in the article "Requirements Engineering: Best Practice" (Fricker, Grau, & Zwingli, 2014).

RE is a very crucial part of a product or project. Many studies show that projects have failed because of poor requirements analysis (The Standish Group, 2014). Even though plenty of state-of-the-art RE techniques exist, adequate implementation and satisfactory execution are obviously still missing. A good overview of frequently observed deficiencies in RE processes is provided in the Trends & Benchmark Reports on Software Development (SwissQ & Gallen, 2014). Typical deficiencies are misunderstandings in communication, continuously growing or changing requirements, or time pressure.

To overcome these inadequacies, new methods of Model-based Requirements Engineering have been developed in recent years. Model-based Requirements Engineering (MBRE) is an approach in which requirements and related business and development information are collected, organized, and structured not only by using natural language, but also with formal, semi-formal, or informal modeling languages (Teufl, Khalil, & Mou, 2013). These models are mostly a reduced and descriptive representation of the subject of discussion compared to the real world. Models in MBRE allow documenting requirements and their relationships to other artifacts in a (mostly graphical) language with less interpretation possibilities than documenting in natural language only would present. Additionally, they can provide abstraction and different perspectives on the data and therefore support communication and discussion among all stakeholders. Modeling with UML or SysML is common practice in MBRE.

5.2.2 Practical Cases

The German Federal Ministry of Defense issued a directive for an efficient and uniform product acquisition and utilization process. This process, named Customer Product Management CPM (Germany Federal Ministry of Defense, 2012), describes procedures for RE, procurement, and in-service support in the German armed forces. It covers the whole system life cycle from the early concept stage in the system life cycle up to retirement on a very high level. The directive defines three main phases in the system life cycle:

- Analysis phase: In the analysis phase, the first objective is to identify capability gaps and to prioritize measures for closing such gaps. In this phase, most of the RE activities and system modeling activities are performed.
- Production phase: In the production phase, the objective is to provide users/operators with suitable and operational products and services in good time. Requirements models and system models are further developed and form the basis of a specification for an award of construction contract.
- In-service phase: The in-service phase covers the use of products and services in accordance with their intended purpose. All measures for maintaining and restoring operational viability, capability, and readiness must be carried out

in order to ensure the safe and economic use of products and services under realistic conditions and in a legal manner until disposal.

Together with the Federal Office of Bundeswehr Equipment, Information Technology and In-Service Support (BAAINBw) as the representative of the German Federal Ministry of Defense, Fraunhofer IESE conducted selected parts of the analysis phase and production phase with a systems engineering approach for a new modular multipurpose combat ship class. Since this project has a very large size and budget and the time schedule is fixed and tight, a pragmatic approach for systems engineering was applied (Webel, et al., 2015).

Analysis Phase part 1: The RE activities in the first part of the analysis phase of the CPM process resulted in a hierarchical catalog of functional requirements prioritized according to their importance or criticality (Prioritized Requirements Catalog). This catalog is based on defined operation and usage conditions and related usage profiles.

- Requirements were formulated in a functional way in order to document only the operational needs and leave open the technical solution.
- Requirements were categorized hierarchically into functional groups according to common performance parameters (such as functionality and operation on the top level).
- Requirements were prioritized in order to identify criteria for stopping the project in case of non-fulfillment.
- Requirements were weighed against each other to document the contribution of each requirement to the organizational capabilities and to separate important and non-important requirements.
- All requirements underwent a quality assurance process in order to fulfill the quality criteria according to IEEE 830:1998 Software Requirements Specification, such as correctness, completeness, consistency, traceability to their origin, verifiability, unambiguity, etc.

Finally, an operational architecture according to the Architecture Data Model of the Armed Forces, an adaption of the NATO architecture framework, version 3.1 (NATO), was developed. This architecture framework is based on SysML and extends the modeling capabilities with domain-specific extensions individually adapted to the needs of defense projects. In the analysis phase part 1, the operational architecture contained mainly

- a capability view model with a high-level understanding of the organizational capabilities with a description of the tasks and activities, operational elements, and information exchange,
- an operational view model that generally reflects requirements or operations from a user's perspective (scenarios), and

- a top-level glimpse at a systems view model with actual or proposed implementations and descriptions of systems, and the system context.

An integrated approach for RE and system modeling, which was specifically developed along with tool chain integration, ensured that the requirements model and the system models were in sync. For example, the functional requirements were translated into system functions mapped to the initial system components defined in the systems view of the architecture model.

Additionally, a style guide helped to specify requirements in a good linguistic style and gave advice on how to comply with the requirements definition procedure and tools.

Analysis Phase part 2: In the second part of the analysis phase in the CPM process, further functional and non-functional requirements were added. The requirements model grew up to more than 10,000 requirements of different types such as technical, logistics, product life cycle, project and risk management, and quality management. A system structure was developed and all requirements were linked to the system structure on the one hand and to the functional requirements model built in the first part of the analysis phase on the other hand in order to ensure traceability.

The architecture model was extended by a system architecture according to the Architecture Data Model of the Armed Forces. The systems view was extended by a much more detailed structure view in synchronization with the system structure of the requirements model. Important system interconnections among system components and between system components and the context were added.

In this phase, the integrated approach for RE and system modeling was further developed to map technical requirements to the system components in the system model and to keep them in sync during the analysis phase.

Production phase: With the beginning of the production phase according to the CPM process, the requirements model and the system architecture were further specified and detailed in negotiation with industry partners with the aim of developing a system specification as a foundation for ordering the realization. This phase is currently ongoing.

Both the directive regarding the definition of requirements and the directive regarding the model-driven design are highly integrated. In all models for requirements, operational architectures, and system architectures, the same functional and non-functional requirements were specified and mapped to the appropriate model elements to ensure traceability in all phases of the system life cycle. To perform this efficiently, a customized tool chain was developed comprising requirements management, system modeling, traceability and impact analysis, prioritization of requirements, and communication and collaboration.

All tools were customized and extended with their integrated programming engines or additional scripts to realize seamless interaction according to the defined processes for RE and system modeling. Additionally, several operation manuals were created and several tutorials were performed for the stakeholders to assist each of the more than 100 stakeholders involved in the CPM processes of the project in collecting the required data with the right tools and using the tools in the right way for processing and forwarding the data to the next step in the process.

5.2.3 Lessons Learned and Recommendations

Based on the lessons learned while collaborating with the customer in this project, the following recommendations can be provided from the Fraunhofer IESE point of view regarding the introduction and usage of RE methods and RE tools in such kinds of industry projects.

- Requirements Elicitation: The selection of an appropriate requirements elicitation technique should depend on the factors influencing the project and especially on the expected availability and skills of the required stakeholders. These factors represent considerable risks that have to be addressed. The RE primer in (SOPHIST, 2016) contains a selection matrix that provides very good hints on which elicitation technique is best suited for which influencing factors.

- Requirements Analysis: The analysis process of refining the users' needs and constraints should be considered in the overall project plan. Refining users' needs and constraints often leads to additional work, where the original stakeholders have to be involved again and have to agree on suggested changes. This process can often be supported very strongly with RE tools by creating specific selections and viewpoints on the requirements that have to be reworked. In the project described above, it was very valuable that the RE tool used was highly configurable and could be extended with scripts to create these specific selections and views.

- Requirements Validation: The process of ensuring completeness, correctness, clearness, and consistency of requirements should be aligned with standardized quality measures such as defined in the standard IEEE:830. This can be significantly supported by using an RE tool that is able to define constraints on individual attributes or can be extended by scripts or programs to check individual project-/product-specific parameters automatically. Reworking requirements when quality deficits are detected should always be foreseen in the project schedule and the original stakeholders should be involved again to resolve the issues.

- Requirements Specification: Documenting the users' needs and constraints clearly and precisely with all necessary attributes can largely be supported by an RE tool. It should be possible to adjust or extend this tool with a scripting engine to build up tool chains in order to provide comprehensive support for all requirements engineering processes.

- Requirements Change Management: Defining project-/product-specific processes related to the requirements engineering processes and using all available tools to support these processes (e.g., analysis tools, import/export tools, modeling tools) is necessary to keep control of all data processed. A collaboration tool specifically configured to support these processes helps to implement this and forces all stakeholders to follow the defined processes. One additional specific finding of the large project described above was that implementing a Change Control Board in the RE processes that checks and releases intended changes on requirements while considering their impact on other project/product artifacts, helps to avoid unintended changes in the requirements database.

5.3 System Verification and Validation

The extensive use of software in technical devices in many embedded systems domains has become the main driver for new innovations. The growing complexity of the software-controlled parts increases the impact of software defects on the quality properties of the integrated system. The effectiveness and efficiency of system verification and validation processes play a crucial role in the development of software and software-intensive systems in order to meet the specified quality requirements and the customer needs.

Innovative model-based quality assurance techniques have been developed to analyze, verify, and validate the different output artifacts of the development activities, including requirements, design models, program code, and integrated electronic control units. Recent trends aim at early quality assurance, such as virtual validation techniques (Feth, Bauer, & Kuhn, 2015) and highly automated verification and validation using mathematical models and appropriate tool chains.

5.3.1 Example Approaches

The Integrated Quality Assurance (InQA) approach is a systematic method for the combination of static and dynamic quality assurance techniques such as different formal verification, review, and testing techniques (Elberzhager, Rosbach, & Bauer, 2014). The combined techniques benefit from each other to exploit synergy effects, i.e., results from one applied quality assurance technique, such as component coverage and defect distribution, is used to guide and fine-tune the subsequent ones in order to improve the quality assurance process.

The InQA approach consists of three main steps: definition, calibration, and application. In the definition step, the objective of the integrated application of the quality assurance techniques and the context has to be defined, such as experience of the test engineers, available effort, or type and maturity of the software being developed. The second step, calibration, is continuously applied to achieve a valid and mature knowledge base for smart integration. First of all, the appropriate quality assurance techniques have to be selected, the order needs to be defined, and the level (e.g., component or system level) of their application must be determined. Then the data and the metrics that should be considered, such as defect numbers, complexity of components, and effort numbers, have to be defined. The third step, application, deals with the validation of assumptions and the evaluation of how this knowledge can be used to exploit synergies during quality assurance activities.

5.3.2 Practical Cases

The applicability and impact of the InQA approach for technical systems from the transportation domains have been assessed in the large-scale European project MBAT (Kläs, Bauer, Dereani, Söderqvist, & Helle, 2015). Within this evaluation project, the InQA approach was tailored and adapted for the early quality assurance of technical systems. The set of techniques comprised the verification and validation of different system design artifacts by means of system simulation, model-based quality assurance, i.e., techniques that work with specific mathematical models for the verification of properties, such as models of the code structure or system composition, and the derivation of functional test cases, such as data flow or behavior models.

In the MBAT project, research partners, tool vendors, and industrial use case providers from 39 organizations and eight countries jointly investigated and developed quality assurance techniques and the corresponding tool platforms for safety-related software-intensive systems from different transportation domains, i.e., automotive, avionics, and rail systems. 13 industrial use cases from different leading-edge companies like Daimler and Volvo were conducted and evaluated.

The use cases stated the context, settings, and problems that should be addressed by the technologies and provided the opportunity to get quantitative feedback by conducting a corresponding case study. The addressed use cases covered different steps of the system and software quality assurance processes as well as different quality properties, such as functional correctness, time behavior, and compliance with standards, e.g. ISO 26262 (ISO, 2011) for passenger cars, to evaluate the range of applicability of the techniques. For example, the automotive use cases by Daimler (a light control subsystem of a passenger car) and Volvo (a brake-by-wire subsystem) dealt with the verification and validation of software design artifacts and program code.

In the evaluation, the relevant goals and assessment aspects of the industrial use cases were determined and refined. The main goals comprised the verification and validation costs, the defect costs, and the system quality. All use cases and the underlying combined quality assurance approaches, including InQA, were assessed regarding these goals in several iterations.

5.3.3 Lessons Learned and Recommendations

The costs for the application of verification and validation techniques could be significantly reduced, by an average of 32% considering the data collected in the 13 case studies. The costs caused by remaining defects in subsequent development stages could also be reduced by an average of 27%. The goals for the system quality could not be aggregated like the cost items due to the variety of the sub-goals assessed, such as test coverage and post-release defects. In the use cases, all sub-criteria of this goal could be improved by at least 8%. The significant improvement of the cost and quality aspects in all use cases are very promising for the future application of combined quality assurance approaches, including InQA.

5.4 Systems Engineering Tool Chain Integration

In industrial practice, a series of method- and tool-related impediments complicate systems engineering in general and model-based systems engineering in particular.

First, there exists a broad heterogeneity of engineering methods, tools, and data involved in the engineering platforms across the life cycle. Second, there is an increasing need to bridge the gap between development platforms and operational ones, for instance in the context of safety-critical systems. By doing so, human in-the-loop or virtual testing, heterogeneous co-simulation, or monitoring and maintenance of large-scale distributed applications can improve the development and decision-making processes in large developing organizations. Third, the distributed and multi-tier nature of development teams in modern large-scale organizations, spread over multiple countries and suppliers, is an issue. The common denominator of these factors is the need to interoperate seamlessly in today's (still fragmented) tool landscapes.

5.4.1 Example Approaches

Across the automotive, aerospace, rail, and health care domains, the CRYSTAL (critical system engineering acceleration) project researched different approaches with broad industry involvement to improve method and tool interoperability in systems engineering. The project achieved valuable results, e.g., an overview of typical engineering methods and tool chains in the different application domains, clearly identified interoperability challenges, an Interoperability Specification (IOS), improved tool interfaces and adapters, and findings related to sound systems engineering use cases (CRYSTAL, 2016).

The key idea promoted in the CRYSTAL project was to rely on standardized integration interfaces to support lifecycle interoperability with the aim of overcoming redundant integration problems across the boundaries of engineering disciplines, application domains, and tool providers.

Such standardized integration interfaces have to define lightweight and generic concepts as a common denominator for all the artifacts used holistically throughout the development cycle. In the context of lifecycle interoperability, the focus is on the semantics of the links and dependencies among the artifacts crossing the boundaries between the engineering disciplines.

The emerging open standard OSLC (Open Services for Lifecycle Collaboration) was taken as a basis to tackle the interoperability problems (Open Services for Lifecycle Collaboration, 2016). OSLC defines a set of specifications focusing on the support of life cycle activities. In the meantime, the OSLC open initiative has grown up from a "loosely coupled" web community to a member of the open standard organization OASIS. Many commercial and open source products have adopted the open standard and the number of participating organizations is constantly growing.

Although OSLC is already an excellent basis for an interoperability specification, some additional needs for interoperability were identified and the following extensions were proposed by the CRYSTAL project:

- OSLC Configuration and Change Management specification,
- OSLC Tracked Resource Set specification, allowing a server to expose an exact set of resources, track additions to and removals from the set, and track changes to the resources in the set.

With these specifications, OSLC is a promising approach to mitigate interoperability challenges in systems engineering and has gained momentum in the tool industry.

5.4.2 Practical Cases

The public use cases of the CRYSTAL project in the aerospace and automotive domains provide good overviews of typical engineering methods, related interoperability issues, and possible tool integrations. The aerospace case focuses on the specification, analysis, design, and simulation of a regional aircraft de-icing system. The process steps covered can be seen in Figure 14.

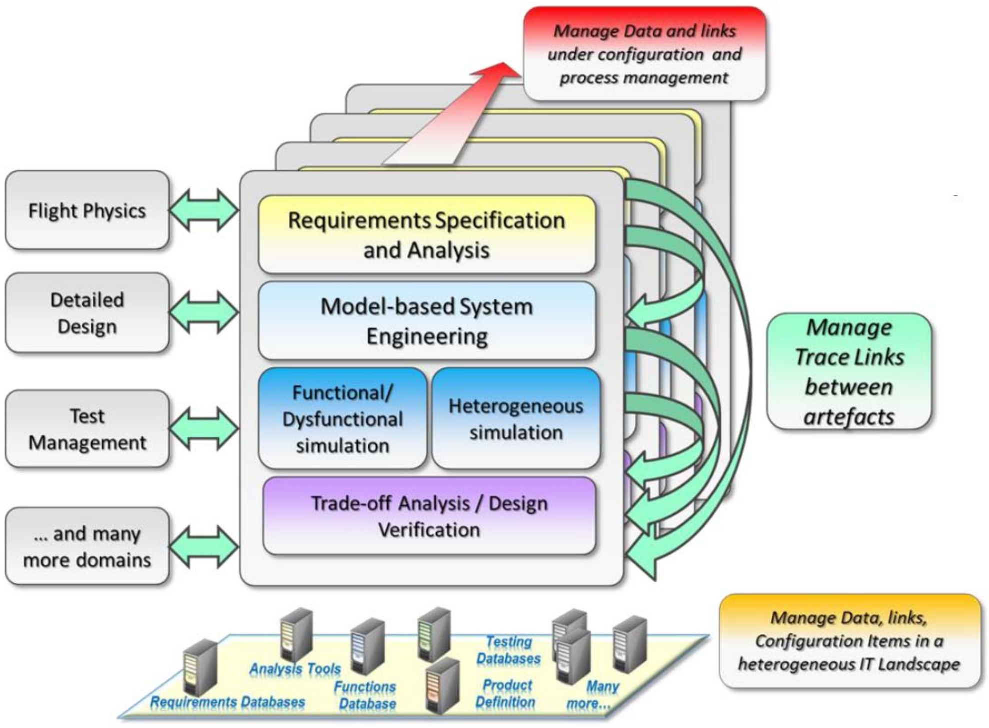
Manage Data and links under configuration and process management
Flight Physics
Detailed Design
Test Management
... and many more domains
Requirements Specification and Analysis
Model-based System Engineering
Functional/ Dysfunctional simulation
Heterogeneous simulation
Trade-off Analysis / Design Verification
Manage Trace Links between artefacts
Requirements Databases
Analysis Tools
Functions Database
Testing Databases
Product Definition
Many more...
Manage Data, links, Configuration Items in a heterogeneous IT Landscape

Figure 14: Process steps covered by the CRYSTAL public aerospace case (Source: Airbus Group)

The automotive case covers different stages in the development of a car – from powertrain design on the vehicle level down to the development of microprocessors and software – and provides insights into the engineering settings of different automotive companies. The case focuses on interoperability challenges arising throughout the entire V-model, including system analysis, variability or variant management, functional safety, and traceability. The covered stages and aspects can be found in Figure 15.

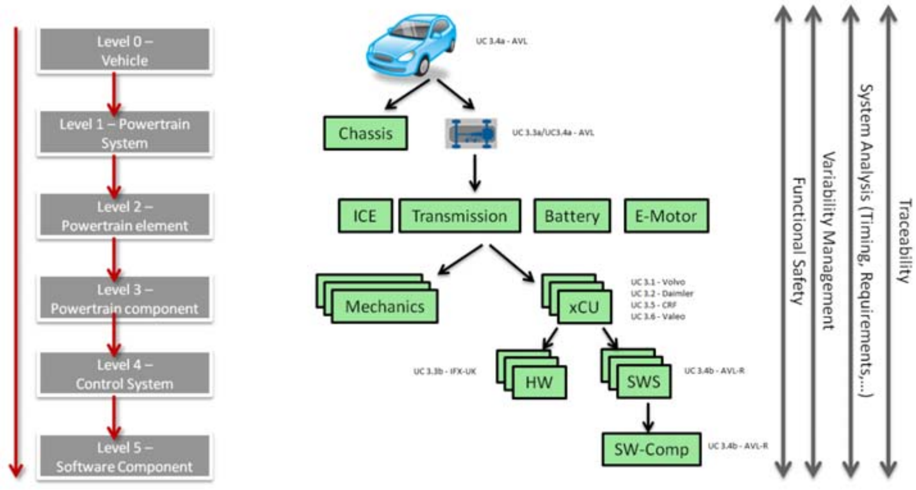

Figure 15: Stages and aspects covered by the CRYSTAL public automotive case (Source: AVL)

The practice areas and tools included in the different use cases comprise the following areas and cover a substantial amount of tool functionality typically used in systems engineering settings:

- Advanced traceability: Reqtify, Rational SSE, RELM
- Model-based systems engineering: RequisitePro, IBM DOORS NG, Rhapsody, Design Manager, PTC Integrity, Modeler
- Requirements quality analysis: Requirements Quality Suite
- Test management: TVS assureSign
- Safety analysis: Fault Tree+
- Simulation: Simulink, OpenModelica
- Variability management: pure::variants
- Process automation: Rational Method Composer, Rational Team Concert

With respect to the interoperability of the tool chains, engineers typically expect the following:

- Semantic links are created across tool boundaries supporting uniform impact and coverage analysis, reporting and metrics.
- Requirements are checked against quality characteristics and improvement is guided.
- System design and functional safety are seamlessly integrated using shared artifacts.
- Simulation models focusing on distinct system aspects can be coupled into a holistic system simulation.
- Variability models manage explicit variation points in the various artifact types and resolve the variabilities of different product variants.

- Workflow support is provided and ensures process compliance.

5.4.3 Lessons Learned and Recommendations

The experience in the use cases shows that tool interoperability can be improved substantially based on OSLC, and has advanced significantly in recent years. An increasing number of tool providers such as IBM, PTC, PureSystems, Siemens, etc., are providing standardized interfaces to interoperate with other tools and have shown compelling tool interoperation scenarios among tools from different tool providers.

Besides these advances, certain complexities regarding the setup and maintenance of the tool adapters and data also became apparent in the use cases. Another issue is link management. For the time being, OSLC does not really specify where links should be managed and how – this is completely up to the developer of the interfaces. Regarding the implementation of OSLC-based interfaces, it became apparent that it is quite challenging to implement an OSLC interface for an existing tool because the availability of the source code is a prerequisite.

A main lesson learned with regard to tool interoperability is that tool interoperation still remains an issue to be investigated in detail on a tool-by-tool basis in a concrete setting. The generalization of meta models and tool interfaces for the typical interoperation scenarios is work in progress and it iss unclear whether this will be achieved at all. Following a use-case-driven approach to improve tool interoperation is a good practice to identify shortcomings and value-adding improvements in the tool chains in a systematic and measurable way. This can also help to educate the engineers in new tooling capabilities.

Beyond tool interoperation, open data formats can also help to archive important data without facing the challenge of having to reinstall complicated tool infrastructures in order to access the data of past projects.

5.5 Virtual Engineering of Systems

Virtual engineering substitutes real artifacts with simulation models. Substituted artifacts may be mechanical parts that are substituted by CAD models, hardware platforms under development that are substituted by virtual platforms that implement instruction set simulators, and software implementations that are substituted, e.g., by Simulink behavior models.

The main purpose of virtual engineering is to ensure important properties and features of artifacts under development. Virtual prototypes are available much earlier than real prototype implementations. Evaluating features with virtual prototypes enables quantitative evaluations much earlier than if implementations/realizations were to be used. This lowers risks when developing complex systems and reduces the effort required for rework because defects or wrong specifications are detected early.

Since complex systems consist of a large number of different artifacts, it is challenging for developers to create simulations that include all relevant aspects. This is, however, necessary in future designs to detect emergence effects, which happen due to the interactions of different parts of systems. One example for the evaluation of emergence effects is the performance evaluation of car-to-car communication, which depends on properties of the wireless networks used, driver behavior, traffic models, and protocol behavior. Simulation tools are usually specialized and have a narrow focus. They only simulate selected effects. Simulation of emergence effects is therefore hard to evaluate without integrating simulators for all relevant parts of the system under development. In the last few years, simulator coupling has been getting popular as a means to overcome this situation.

5.5.1 Example Approaches

Developing simulator couplings is difficult because the models of computation and communication (MOCCs) of simulation models often differ. These need to be integrated consistently (Kuhn, Forster, Braun, & Gotzhein, 2013).

An MOCC defines when a simulation model is executed and how it communicates. Three common MOCCs are Discrete Time, Discrete Event, and Continuous Time models. One common application in industry is the virtual evaluation of Electrical/Electronic (E/E) architectures. E/E architectures consist of hardware control units that are connected by networks. Different networks are connected via gateways. Tasks are deployed to electronic control units. Depending on the implemented control algorithms, a task might be sensitive with respect to scheduling and communication delays, or sensitive to communication and scheduling jitter. The decision about whether one task is to be deployed on a specific processor of a particular controller might therefore significantly affect the performance of an algorithm. Furthermore, network communication and safety mechanisms can be evaluated using the virtual E/E architecture.

5.5.2 Practical Cases

The following application example is an anonymized result from an industry project. The project was about the development of a safety mechanism for a remotely controlled lift that is attached to a truck. The lift should be controlled with a smartphone, which is considered to be an unsafe device. The safeguarding mechanism was to be implemented on a gateway hardware in the truck, while the smartphone was treated as a simple sensor. The safety mechanism consisted of a UI concept on the smartphone that ensured that the measured user intensions were unambiguous, a communication protocol on the smartphone that implemented a safe communication layer, the communication layer on the gateway device, and a logic that evaluated the received results. This logic needed to ensure that the measured sensor inputs represented the user's intensions, i.e., that no sensor defects were masking valid user inputs, that no transmission errors occurred, and that no conflicting commands were received.

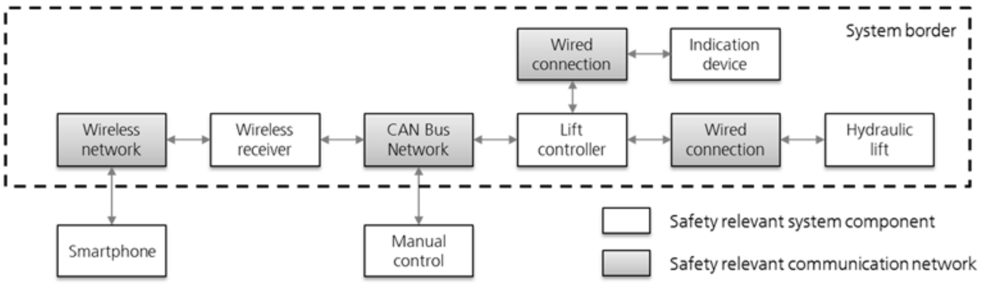

Figure 16: Application example for virtual evaluation

The virtual evaluation had to check that the gateway logic yielded safe system behavior in all cases. This was implemented by defining a number of test cases (scenarios) that yielded correct behavior, and by defining rules describing safe system behavior. The simulation injects faults according to fault models, which include, for example, bit flips in communication networks or stuck-at faults of sensors. In all cases, the system must either yield the intended behavior of the users or one of the predefined safe behaviors. In the case of the hydraulic lift, this was a stop of all movements.

5.5.3 Lessons Learned and Recommendations

By using virtual engineering, it is possible to also develop revolutionary concepts that are not an evolution of existing approaches, but rather realize new ideas. Simulations enable developers to collect experience and to quantitatively evaluate and compare the performance of different approaches. The ability to evaluate critical aspects early and without risks in simulation in conjunction with the increasing speed and accuracy of simulation models continuously increases the importance and applicability of virtual engineering techniques.

In the near future, product complexity will significantly increase and system development will become more multi-disciplined. As a consequence, the assembly of large systems will become more difficult and cost-intensive.

Hardware-in-the-Loop testing is a common practice in industry today. However, integration testing is performed at a very late project stage. The correction of defects is therefore very costly, leads to complications and to unnecessary project delays. Considering the increasing system complexity and architecture, integration testing should start as soon as possible in the development process. Virtual

Hardware-in-the-Loop testbeds, created by coupling existing simulators, should be considered as an efficient approach.

Bibliography

acatech - National Academy of Science and Engineering. (2016, 9 23). Retrieved from http://www.acatech.de

Achary, B., & Actis, M. (2013). Introducing the CTA confept. *Astroparticle Physics, 43*(3).

Bruner, J. (2013). *Industrial Internet.* O'Reilly Media, Inc.

Cavalcante, E., Pereira, J., Alves, M., Maia, P., Moura, R., Batista, T., . . . Pires, P. (2016). On the interplay of Internet of Things and Cloud Computing: A systematic mapping study. *Computer Communications*.

Conforto, E., Rossi, M., Rebentisch, E., Oehmen, J., & Pacenza, M. (2013). *Survey Report: Improving Integration of Program Management and Systems Engineering.* Philadelphia: PMI and INCOSE.

CRYSTAL. (2016). Retrieved from CRYSTAL - Critical System Engineering Acceleration: http://www.crystal-artemis.eu

Ebert, C. (2014). Requirements Engineering - Industry Practice. Vector Consulting Services. Retrieved 09 08, 2016, from http://vector.com/portal/medien/vector_consulting/publications/Ebert_RequirementsEngineering_Overview_EN.pdf

Elberzhager, F., Rosbach, A., & Bauer, T. (2014). An Integrated Analysis and Testing Methodology to Support Model-Based Quality Assurance. *Software quality days (SWQD 2014).* Vienna: Springer.

Elm, J. P., & Goldenson, D. R. (2012). *The Business Case for Systems Engineering Study: Results of the Systems Engineering Effectiveness Study.* Carnegie Mellon University, Software Engineering Institute, AESS, NDIA.

Estefan, J. A. (2007). *Survey of Model-Based Systems Engineering (MBSE) Methodologies.* INCOSE MBSE Focus Group.

Federal Ministry for Economic Affairs and Energy. (2016, 9 23). *Plattform Industrie 4.0*. Retrieved from http://www.plattform-i40.de

Feth, P., Bauer, T., & Kuhn, T. (2015). Virtual Validation of Cyber Physical Systems. *Software Engineering and Management 2015 (SE 2015)* (pp. 201-206). Dresden: Springer.

Fricker, S., Grau, R., & Zwingli, A. (2014). Requirements Engineering: Best Practice. In S. A. Fricker, C. Thümmler, & A. Gavras, *Requirements Engineering for Digital Health* (pp. 25-38). Heidelberg, New York, Dordrecht, London: Springer.

Gausemeier, P.-I., Dumitrescu, R., Steffen, D., Czaja, A., Wiederkehr, O., & Tschirner, C. (2015). *Systems Engineering in industrial Practice.* Heinz Nixdorf Institute, Fraunhofer Institute for Production Technology, Unity AG.

Germany Federal Ministry of Defense. (2012). *Customer Product Management (amended).* German Federal Ministry of Defense.

Hankel, M.; Bosch Rexroth. (2015). *Industrie 4.0: The Reference Architectural Model Industrie 4.0 (RAMI 4.0).* Frankfurt am Main, Germany: ZVEI - German Electrical and Electronic Manufacturers' Association.

Heidrich, J., Trendowicz, A., & Ebert, C. (2016). Exploiting Big Data's Benefits. *IEEE Software, 33*(4), 111-116.

IBM. (n.d.). IBM Rational Harmony for Systems Engineering: The Harmony Process. IBM. Retrieved 09 22, 2016, from http://www.ibm.com/support/knowledgecenter/SSB2MU_8.1.5/com.btc.tcatg.user.doc/topics/atgreqcov_SecSysControllerHarmony.html

ISO. (2011). *ISO 26262: Road vehicles – Functional safety.* International Organization for Standardization.

Jazdi, N. (2014). Cyber physical systems in the context of Industry 4.0. *IEEE International Conference on Automation, Quality and Testing, Robotics* (pp. 1-4). IEEE.

Kagermann, H., Helbig, J., Hellinger, A., & Wahlster, W. (2013). *Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry.* Final report of the Industrie 4.0 Working Group, Forschungsunion. Retrieved from http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Material_fuer_Sonderseiten/Industrie_4.0/Final_report__Industrie_4.0_accessible.pdf

Kläs, M., Bauer, T., Dereani, A., Söderqvist, T., & Helle, P. (2015). A Large-Scale Technology Evaluation Study: Effects of Model-Based Analysis and Testing. *37th International Conference on Software Engineering (ICSE 2015)* (pp. 119-128). Stockholm: IEEE Computer Society.

Kuhn, T., & Antonino, P. (2014). Model Driven Development of Embedded Systems. *Embedded Software Engineering Kongress (ESE).*

Kuhn, T., Forster, T., Braun, T., & Gotzhein, R. (2013). FERAL – Framework for Simulator Coupling on Requirements and Architecture Level. *ACM-IEEE International Conference on Formal Methods and Models for System Design.* Portland, USA.

Lee, J., Lapira, E., Bagheri, B., & Kao, H. (2013). Recent advances and trends in predictive manufacturing systems in big data environment. *Manufacturing Letters, 1*(1), 38-41.

Mell, P., & Grance, T. (211). *The NIST Definition of Cloud Computing.* Gaithersburg, Maryland, USA: National Institute of Standards and Technology.

Minerva, R., Biru, A., & Rotondi, D. (2015). Towards a definition of the Internet of Things (IoT). *IEEE Internet Initiative*.

Naab, M., Knodel, J., Kuhn, T., & Rost, D. (2016). *Smart Ecosystems Reference Model.* Fraunhofer IESE.

NATO. (n.d.). *NATO Architecture Framework RFCP Regarding NAF V3.1 Chapter 5: NATO Architecture Framework Metamodel (NMM) and Architecture Data Exchange Specification (ADES).* Retrieved from http://www.nhqc3s.nato.int/ARCHITECTURE

Open Services for Lifecycle Collaboration. (2016). Retrieved from http://open-services.net

Oya, I., Füßling, M., Oliveira Antonino, P., Conforti, V., Hagge, L., Melkumyan, D., . . . the CTA consortium. (2016). The Software Architecture for the Cherenkov Telescope Array. *SPIE - International Society for Optics and Photonics Conference and Exhibition 2016. 9913.* Edinburgh, Scotland, UK: Software and Cyberinfrastructure for Astronomy III.

Pisching, M., Junqueira, F., dos Santos Filho, D., & Miyagi, P. (2015). AN ARCHITECTURE FOR ORGANIZING AND LOCATING SERVICES TO THE INDUSTRY 4.0. *ABCM International Congress of Mechanical Engineering.* Rio de Janeiro, Brazil.

Pohl, K., Achatz, R., & Broy, M. (2012). *Model-Based Engineering of Embedded Systems- The SPES 2020 Methodology.* Springer.

Software Engineering Institute, Carnegie Mellon. (n.d.). A Framework for Software Product Line Practice - Requirements Engineering. Retrieved 09 08, 2016, from http://www.sei.cmu.edu/productlines/frame_report/req_eng.htm

SOPHIST. (2016). RE Primer. Retrieved 09 21, 2016, from https://www.sophist.de/publikationen/wissen-for-free/

Sophist. (n.d.). FAQ Requirements Engineering. Retrieved 09 08, 2016, from https://www.sophist.de/en/requirements/requirements-engineering/faq-requirements-engineering/

SwissQ, & Gallen, U. o. (2014). Trends & Benchmarks Report in Software Development. Zürich: SwissQ Consulting AG. Retrieved 09 08, 2016, from http://swissq.it/wp-content/uploads/2016/02/Agile_RE_Testing-Trends_und_Benchmarks2014.pdf

Teufl, S., Khalil, M., & Mou, D. (2013). Requirements for a Model-based Requirements: Systematic Literature Review and Survey. Munich: fortiss GmbH. Retrieved 09 08, 2016, from http://download.fortiss.org/public/projects/af3/research/2013/MbRE_tool_requirements_for_embedded_systems.pdf

The Standish Group. (2014). *CHAOS Report.* The Standish Group.

Wang, H., Osen, O., Li, G., Li, W., Dai, H., & Zeng, W. (2015). Big data and industrial internet of things for the maritime industry in northwestern norway. *IEEE Region 10 Conference TENCON* (pp. 1-5). IEEE.

Webel, C., Darting, S., Schmitt, M., Kleinberger, T., Braun, R., & Weber, J. (2015). Pragmatisches Systems Engineering in einem Großprojekt mit Einschränkungen. *Tag des Systems Engineering 2015* (pp. 323-332). Munich: Carl Hanser Verlag GmbH & Co. KG.

Document Information

Title:	WP2/D2.1: Systems Engineering Study: Challenges and Best Practices
Date:	October 12, 2016
Status:	Final Reviewed 2