Otto-von-Guericke-Universität Magdeburg

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

MB FAKULTÄT FÜR MASCHINENBAU

Model-driven Basic Engineering of flexible Production Systems with the RAMI Toolbox

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieur

(Dr.-Ing.)

von

DI Christoph Binder, M.Sc.

geboren am 25.04.1992 in Innsbruck

genehmigt durch die Fakultät für Maschinenbau
der Otto-von-Guericke-Universität Magdeburg

Gutachter:	apl. Prof. Dr.-Ing. habil. Arndt Lüder
	Prof. Dr.-Ing. Christiane Beyer
	Prof. (FH) Dr. Christian Neureiter
Kommissionsvorsitz:	Prof. Dr.-Ing. habil. Thorsten Halle

Promotionskolloquium am 17.01.2024

Contents

List of Figures

List of Tables

List of Acronyms

AAS	asset administration shell
ADL	architecture description language
ADSRM	agile design science research methodology
AHP	analytic hierarchy process
AI	artificial intelligence
API	application programming interface
BPMN	business process model and notation
BUC	Business Use Case
CAEX	computer-aided engineering exchange
CIM	Computational Independent Model
CPS	cyber-physical system
DLL	dynamic-link library
DSL	domain-specific Language
DSR	Design Science Research
DSSE	domain-specific systems engineering
EA	Enterprise Architect
ESE	enterprise systems engineering
ETFA	Emerging Technologies and Factory Automation
FAS	Functional Architecture for systems
FREDOSAR	Free Educational Open System Architecture
GPL	general purpose language
GUI	graphical user interface
HCI	human-computer interaction
HLUC	High-Level Use Case
HMI	human machine interface
ICT	information and communication technologie
IIoT	industrial internet of things
IIRA	Industrial Internet Reference Architecture
INCOSE	International Council on Systems Engineering

INDIN	Industrial Informatics
IoE	internet of everything
IoT	internet of things
JSON	JavaScript Object Notation
KPI	key performance indicator
M2M	machine to machine
MBE	model-based engineering
MBSE	model-based systems engineering
MDA	model-driven architecture
MDD	model-driven development
MDE	model-driven engineering
MDG	model-driven generation
MDSD	model-driven software development
MQTT	message queuing elemetry transport
NFC	near-field communication
OMG	Object Management Group
OPC UA	Open Platform Communications Unified Architecture
PIM	Platform Independent Model
PLC	programmable logic controller
PoC	proof-of-concept
PPR	product, process and resource
PSI	Platform Specific Implementation
PSM	Platform Specific Model
RAMI 4.0	Reference Architecture Model Industrie 4.0
RFID	radio frequency identification
RM-ODP	Reference Model of Open Distributed Processing
RM-SA	Reference Model for Service Architecture
RTE	round-trip engineering
SAAM	Software-Architecture Analysis Method
SCADA	supervisory control and data acquisition
SDN	software-defined networking
SGAM	Smart Grid Architecture Model
SoA	service-oriented architecture
SoaML	Service-oriented Architecture Modeling Language
SoI	system of interest
SoS	system of systems
SPES	Software Platform Embedded Systems
SysML	Systems Modeling Language

TOGAF	The Open Group Architecture Framework
UAF	Unified Architecture Framework
UML	Unified Modeling Language
VBA	Visual Basic for Applications
XMI	XML metadata interchange
XML	extensible markup language

Kurzfassung

Der Begriff „Industrie 4.0“ beschreibt den Trend von aufkommenden Technologien im industriellen Bereich sowie neuen Automatisierungsmöglichkeiten, welche herkömmliche Produktionslinien revolutionieren. Angetrieben durch sogenannte Cyber-Physische Systeme (CPS) bringt diese Transformation jedoch eine stetig steigende Komplexität mit sich. In weiterer Folge führt die steigende Anzahl dynamisch miteinander verbundener Elemente in industriellen Produktionslinien daher zu einer stetigen Umformung von komplizierten Systemen hin zu komplexen Systemen von Systemen (SoS). Aufgrund der damit einhergehenden Herausforderungen stoßen herkömmliche Engineering-Methoden bei der Entwicklung derartiger Systeme an ihre Grenzen. Da unterschiedliche Ansätze jedoch lediglich versuchen, eine Lösung für einzelne Aspekte zu finden oder auf kleinräumige Bereiche dieser Problemstellung abzielen, wird die Notwendigkeit einer ganzheitlichen Methodik immer offensichtlicher. Hier hat sich modellbasiertes Systems Engineering (MBSE) als Schlüsselfaktor erwiesen, um die unterschiedlichen Interessen aller Beteiligten in einer solchen umfangreichen Infrastruktur zu vereinen und der oben genannten Systemkomplexität entgegenzuwirken.

Daraus schlussfolgernd lautet die Forschungsfrage dieser Arbeit wie folgt: “Wie können domänenspezifische Besonderheiten und modellbasierte Engineering-Konzepte zu einem umfassenden Modellierungsansatz konsolidiert werden, welcher ein ganzheitliches und interdisziplinäres Engineering aktueller und zukünftiger Fertigungssysteme ermöglicht und dabei alle Interessengruppen berücksichtigt?” Um dies zu erreichen, wird zunächst eine breit angelegte State-of-the-Art-Recherche durchgeführt, welche einerseits vielversprechende domänenspezifische Ansätze analysiert und andererseits aktuelle Methoden des MBSE für deren Anwendung untersucht. Basierend auf den Ergebnissen dieser Forschung werden Entwicklungsartefakte definiert und zu einem vollständigen Framework akkumuliert, welches jede Phase des Systems-Engineering-Lebenszyklus adressiert. Aufgrund der Unvorhersehbarkeit zukünftiger Anwendungen und der hohen Änderungsrate in einer solch agilen Domäne werden die Konzepte der Agile Design Science Research Methodology (ADSRM) für den Einsatz in diesem dynamischen Szenario ausgewählt. Die jeweiligen Ergebnisse werden dabei durch die Anwendung unterschiedlicher Fallbeispiele aus der Praxis und deren Evaluierung bewertet. Dabei widerspiegelt die sogenannte RAMI Toolbox das primäre Resultat dieser wissenschaftlichen Abhandlung, welches deren jeweiligen Teilergebnisse in einem Tool zusammenfasst und bereitstellt. Basierend auf jenen Ergebnissen dieser Studie könnte das entwickelte Modellierungs-Framework produzierenden Unternehmen neue Möglichkeiten bieten, um Automatisierungspotenziale zu finden, aktuelle Systeminfrastrukturen zu evaluieren oder internationale Kooperationen einzugehen.

Abstract

Emerging technologies in the industrial area and new automation possibilities promoted by so-called cyber-physical systems (CPS) lead to a constantly increasing complexity in manufacturing systems nowadays, summarized under the term “Industry 4.0”. As a further consequence, the rising number of dynamically interconnected elements in industrial production lines results in a steady transformation from complicated systems to complex systems of systems (SoS). Due to the upcoming challenges accompanied by this trend, conventional engineering methods reach their limits when engineering this kind of system. However, the need for a holistic methodology becomes apparent with varying approaches only trying to find a solution for single aspects or to target small-scaled areas of this problem statement. As model-based systems engineering (MBSE) has proven to be a key enabler when it comes to combining the different interests of all stakeholders in such critical infrastructure, the used concepts are considered to be a suitable method for approaching the aforementioned system complexity.

Thus, the research question of this thesis is: “How can domain-specific particularities and model-based engineering concepts be consolidated into a comprehensive modeling approach enabling holistic and interdisciplinary engineering of current and future manufacturing systems by addressing all stakeholder concerns?”. To achieve this, state-of-the-art research, including a summary of practical domain-specific approaches on the one hand and current and contemporary methods used in MBSE on the other hand, is conducted. Based on the results of this research, the development artifacts are defined and accumulated to a complete framework addressing each phase of the systems engineering life-cycle. Due to the unpredictability of future applications and the high rate of change in such an agile domain, the Agile Design Science Research Methodology (ADSRM) concepts are used in this dynamic scenario. The individual results are thereby assessed by an observational evaluation based on different real-world case studies. However, the so-called RAMI Toolbox reflects the primary outcome of this scientific work, which encapsulates the individual results within a single tool. The developed approach leads manufacturers to find new automation potential, evaluate current system infrastructures or enable the possibility of international collaborations.

Declaration of Honor

"I hereby declare that I produced this thesis without prohibited external assistance and that none other than the listed references and tools have been used. I did not make use of any commercial consultant concerning graduation. A third party did not receive any nonmonetary perquisites neither directly nor indirectly for activities which are connected with the contents of the presented thesis.

All sources of information are clearly marked, including my own publications.

In particular I have not consciously:

– Fabricated data or rejected undesired results
– Misused statistical methods with the aim of drawing other conclusions than those warranted by the available data
– Plagiarized data or publications
– Presented the results of other researchers in a distorted way

I do know that violations of copyright may lead to injunction and damage claims of the author and also to prosecution by the law enforcement authorities. I hereby agree that the thesis may need to be reviewed with an electronic data processing for plagiarism. This work has not yet been submitted as a doctoral thesis in the same or a similar form in Germany or in any other country. It has not yet been published as a whole."

Salzburg, 14.02.2023 Christoph Binder

Chapter 1

Introduction

In recent years, a novel industrial revolution emerged, forcing organizations to adjust production processes and consider new business models to remain competitive. This revolution, summarized under the term “Industry 4.0”, expresses the transformation of traditional production systems towards intelligent value creation networks (Iglesias et al., 2019). This ongoing trend’s main objective is to ensure partly or fully autonomous and flexible manufacturing sustainably. Accompanied by some significant consequences, like factory automation, ubiquitous information management, or flexible production (Ghobakhloo, 2018), the fourth industrial revolution is mainly promoted by the digitalization of analog technologies currently used in manufacturing. Thus, the remainder of this chapter outlines the characteristics of this trend and the resulting needs for this system transformation, reflecting the motivation and scope of this thesis.

1.1 Problem Context

The leading technology drivers encouraging the trend mentioned above originate from the areas of the industrial internet of things (IIoT) as well as cyber-physical systems (CPSs). CPSs represent mostly intelligent system components, which could be considered as systems themselves. In more detail, this enables the automation of industrial production processes by making independent and decentralized decisions, which result in the best possible outcomes for individual scenarios (Rajkumar et al., 2010). Additionally, as a further result, the original product-oriented manufacturing is replaced by technology-based services (Claude Jr and Horne, 1992). Thus, when referring to increasingly adaptive systems within the industrial domain by describing the generic capabilities of its components, the established division of production entities into product, process and resource (PPR), including skills, needs to be introduced (Pfrommer et al., 2013).

In conclusion, such IIoT-based systems result in having a more considerable diversity or scale due to extensive interconnections. This means before actually implementing novel methods enabling efficient production, ubiquitous information exchange, and supporting intelligent decision making (Villalonga et al., 2020), currently used manufacturing systems need to be analyzed and transformed to fulfill the upcoming requirements (Dall'Ora et al., 2021). Usually, this transformation is a slow procedure, as replacing the production system as a whole appears to stop production at once. To remain competitive, manufacturers' production systems must be constantly adapted.

Consequently, transforming original manufacturing systems into intelligent value-creation networks is inevitable. Nevertheless, transforming their manufacturing networks is a big obstacle for most manufacturers, as currently used document-based approaches do not provide the proper tools to combine product design, production system, and manufacturing process (Lüder et al., 2020). In addition, these documents are often contradictory, incomplete, or outdated. Since this transformation is usually expensive and risky, planned changes should be validated, and risks should be elaborated and minimized (Calà, 2019). This can also be derived from currently heightened challenges of intelligent industrial systems, which also were recognized more than 30 years ago (Glasmeier, 1991) and are supported by more recently elaborated ones (Khan et al., 2020):

- Multi-discipline collaboration and early verification are essential.
- No available modeling methodology to track and handle intelligent manufacturing systems exists.
- A consistent and unambiguous system representation is necessary to ensure integrity.

Another obstacle impeding this transformation is the increasing complexity of current manufacturing systems. Falling back to the fact of intelligent internet of things (IoT) devices, which are continuously integrated into production networks, being systems themselves, a comprehensive production network needs to be classified as a system of systems (SoS). The main characteristics to determine this classification can be underpinned by the operational independence, geographic distribution, and evolutionary and emergent behavior of contemporary manufacturing systems (Maier, 1998). In addition, regarding the classification scheme observable in Figure 1.1, as proposed by Haber-

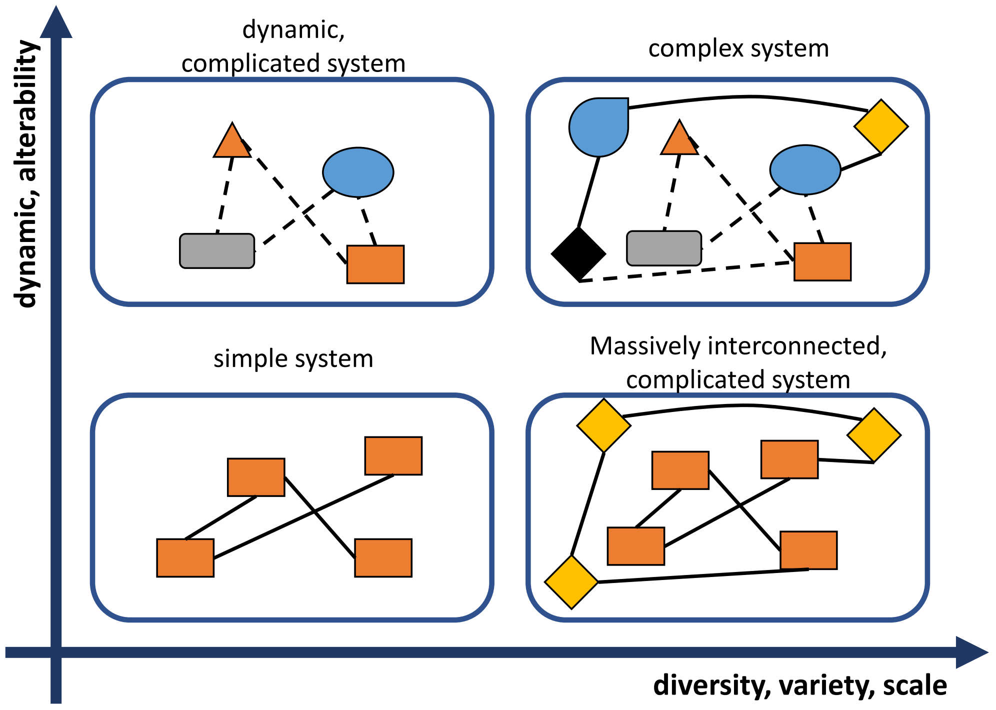

dynamic, alterability
dynamic, complicated system
complex system
simple system
Massively interconnected, complicated system
diversity, variety, scale

Figure 1.1: Systems complexity classification based on (Haberfellner et al., 2015)

fellner et al. (2015), such a system needs to be considered a complex system, while original production lines fall under the criteria of complicated systems. While complicated systems are either large-scale and diverse or exhibit dynamic behavior, complex systems include both of these classification properties. In more detail, on the one hand, a production network consists of a large variety of interconnected IoT devices; on the other hand, production processes need to adapt flexibly, enabling individual production in lot size 1.

Those peculiarities imply that future systems engineering, in particular the integrative development of products, production process, and production resource, will become a difficult task to be achieved. Hence, for the above reasons, it can be said that conventional engineering methods, especially those based on document-based information exchange, reach their limits when being applied in disciplines such as industrial systems engineering (Friedenthal and Oster, 2017). The main reason for exhibiting the limitations of conventional methodologies and simultaneously providing solutions for novel approaches to counteract this issue can be determined with the fourth industrial revolution, better known as Industry 4.0.

1.2 Industry 4.0

As shown in the right column of Design Science Research (DSR) in information systems (Hevner and Chatterjee, 2010), an already existing knowledge base can be applied for the development of design artifacts. This knowledge base constantly changes and expands with advances from industrial projects or scientific research. Thus, new theories, frameworks, and methodologies should be researched and examined by using this research methodology to apply existing knowledge in research projects. For this reason, a detailed overview of existing approaches and fundamental methods is given in this chapter. A brief introduction explains the principle of Industry 4.0 and the associated opportunities and difficulties.

Initially emerged at the Hannover Messe 2011, the term “Industry 4.0” largely left a mark in the German industry. Numerous discussions took place in various sectors, such as research, politics, and the energy sector, regarding how the trend should be understood in detail. What is undisputed, however, is that a large proportion of the issues associated with Industry 4.0 will inevitably occur sooner or later (Drath and Horch, 2014). In their publication, the authors introduced three hypotheses, which facilitate understanding of the concept of CPSs and more closely examine assumptions that are likely to occur. Briefly summarized, these hypotheses deal with topics such as an affordable communication infrastructure for automatic data exchange and the integration of intelligent components in value-creation networks. These components learn functionalities based on the evaluated data and thus make decentralized decisions. However, Drath and Horch (2014) also point out that, at the time of their publication, visions represent a large part of the ideas of Industry 4.0. One of the main drivers behind Industry 4.0 is stated to be flexibility, which was at its peak during manual labor.

Figure 1.2 overviews the four industrial revolutions and the main trigger causing Industry 4.0. Previously to the first industrial revolution, manual labor dominated production. However, in the context of the first revolution, mechanization was introduced, which offered new types of energy sources and led to the replacement of agriculture. However, with the emergence of electricity, gas, and oil, more potential has been made available by engines, ultimately leading to automobiles and planes. The third industrial revolution has been promoted by digitization, the rise of electronics, and telecommunications. Different outcomes in research and development resulted in the proposal of Industry 4.0. (iED Team, 2019)

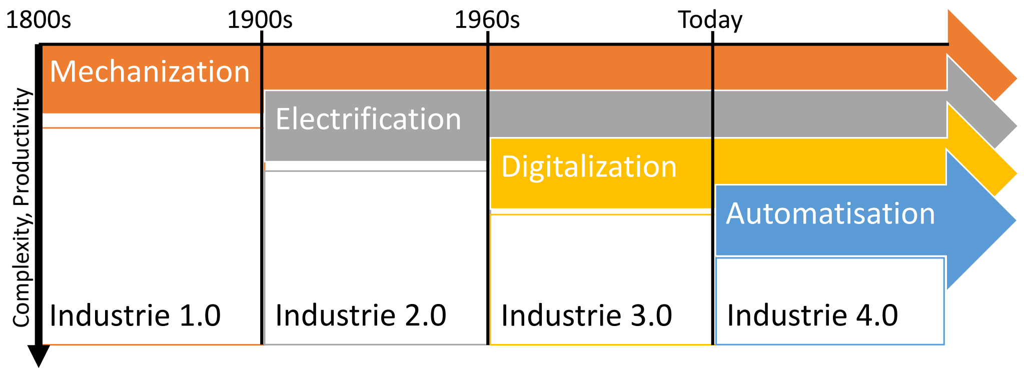


Figure 1.2: Overview and interrelations of the four industrial revolutions based on (Drath and Horch, 2014)

The phenomenon of Industry 4.0 thus describes a protracting change in industrial production to automate the entire value-add process. Results from research and industry in the area of the IIoT are increasingly helping to achieve this goal. By the term Industry 4.0, however, only the trend towards increasing digitalization of industrial production is indicated; a cross-industry definition is deliberately omitted. Therefore, in the first five years after the emergence of the trend, the publishers of various documents tried to classify the term. Based on these different definitions, some researchers have analyzed, filtered, and processed the essential design principles (Hermann et al., 2016). The results of the resulting quantitative text analysis and qualitative literature review include four principles:

Mutual dependency

By gaining new insights, the environment of the IoT has evolved into a broader concept. The so-called internet of everything (IoE) also spans subject areas such as people, content, ideas, and aspects already included by IoT. This means collecting and analyzing all available data as far as possible is crucial. To realize this problem, the involved components need to be scalable on the one hand without sacrificing performance on the other hand. (de Santos and Villalonga, 2015)

Depending on the involved units, a distinction between different types of collaboration is made. This is how communication between homogeneous participants such as people or machines (machine to machine (M2M)) is implemented. Nevertheless, humans can also communicate with machines (human-computer interaction (HCI)).

When it comes to improvement, many companies focus on the shop floor level, where production or logistics departments are located. However, there is greater potential for optimization in the industrialization of business processes. In conclusion, the communication options supported by CPSs might be used to simplify processes or support decision-making. (Schuh et al., 2013)

Information transparency

To make appropriate decisions, all participants in the communication network must have information corresponding to the regarded context. Each component utilizes this information and interprets it to align subsequent actions accordingly. The data originate from different sources such as product manufacturing processes, technical documentation, or operational production information. Therefore, a uniform context should be created, which can be achieved through standardization. In addition, it is also essential that generated raw data, such as sensor recordings, are combined into higher-quality information to be interpreted by other CPSs. (Gorecky et al., 2014

Decentralized decision-making

Due to the lack of standardization in Industry 4.0, decisions are made at the shop floor level. This allows for faster response and better use of product-specific knowledge. However, to increase synergies, data should be centralized and processes modeled globally. To avoid bottlenecks, the individual units must be given more freedom of decision. Nowadays, people mainly make decisions; in the future, CPS will take over this part. As a result, self-optimizing and knowledgeable value-added processes can be set up. (Brettel et al., 2014)

Technical support

CPS not only communicate with each other but also with the physical world. This data exchange offers several advantages, such as energy efficiency, security, and real-time behavior. They can work independently through intelligent networks with existing dependencies. Within the value creation process, these systems should have a supporting effect and be able to react to structures created by people. A single unit works semi-autonomously and thus carries out certain tasks independently. Thus, the higher-level task is determined depending on the situation; the sub-processes are carried out fully automatically. This behavior depends on several factors that ensure correct operation. The higher-level overall system must be stable, and information must be passed on to the outside in a reliable and timely manner. An interface is defined for communication with the system, which can be understood and operated by a human. In summary, it must be ensured that the verification and validation of a CPS is not a one-off event but must be ensured over the entire life cycle. (Rajkumar et al., 2010)

A few years later, a qualitative-quantitative text analysis was carried out again to precisely analyze documents in Industry 4.0. The result is published in (Ghobakhloo, 2018). In addition to technology trends, some design principles were developed, adding a few aspects to those previously mentioned. These include virtualization to create a digital twin of a complete value chain. Another point addressed is system integration, which describes combining the individual system components into a SoS to achieve the desired functionality. The latest research also attempts to identify maturity models from current publications and compare them with the design principles that have also been recognized (Dikhanbayeva et al., 2020). In the mentioned work, the individual models are evaluated based on the degree of compliance with the design principles. However, it can be seen that these principles have barely changed in recent years.

As seen from the extensive literature analyses, the topics to be dealt with in this work can be found in many documents. Many design principles developed over the past ten years dealing with the exchange of information in industrial systems and their implementation through the technical integration of individual CPSs. However, it can also be seen that the complexity of industrial systems is constantly increasing. As a result, practical applications lag behind the theoretically published approaches. This impedes a novel approach, which provides an application framework to any interested practitioner by ensuring the application of theoretically published concepts, needs to emerge, which is the primary motivation for accomplishing this thesis.

1.3 Motivation and Scope

The Reference Architecture Model Industrie 4.0 (RAMI 4.0) (ZVEI - Zentralverband Elektrotechnik- und Elektronikindustrie e.V., 2015) has been developed to provide a standardized approach that tries to counteract the increasing complexity of current or future manufacturing systems by distinguishing into manageable units or summarizing topic-related aspects, which supports the execution of the respective engineering disciplines. In addition, the associated implementation strategy of RAMI 4.0 furthermore explains that systems engineering is a crucial subject area remaining in the focus of research when it comes to handling the complexity of such an industrial system (Bitkom et al., 2015). By doing so, the three-dimensional model falls back on the modeling paradigms *separation of concerns* as well as *divide & conquer*. This means when utilizing RAMI 4.0, instead of generating comprehensive documentation, a consistent architectural model of the industrial system should be created and used for different purposes, such as system optimization or stakeholder communication. In summary, this reference architecture appears to be one of the most promising approaches when dealing with the complexity of current and future industrial systems. However, although being standardized within the DIN SPEC 91345 since 2016, the application of RAMI 4.0 is still widely unexplored (Contreras et al., 2017). It might be that the underlying foundations are primarily theoretical and look good on paper, but their utilization seems to be too abstract for actual implementation.

On the other hand, in terms of applicability, model-based systems engineering (MBSE) appears to be the most promising methodology dealing as a pre-requisite for intelligent engineering products or innovative production systems (Brusa, 2018). As introduced by the International Council on Systems Engineering (INCOSE), it offers a method for engineering such systems using models as significant artifacts. This enables top-down engineering, starting with requirements and finishing with the detailed system design. Additionally, as such manufacturing systems are rarely defined from scratch, MBSE provides a central model dealing as a link between all system stakeholders and offering a documented artifact enabling the ongoing system transformation. Hence, MBSE appears to be one of the most promising approaches for the basic engineering of future production systems (Biffl et al., 2017). Using various models to address all the stakeholders and their concerns reduces difficulties during entry into systems engineering and the anticipated system transformations. However, at the current point of view, industrial applications of MBSE are still widely unexplored, in particular regarding providing a first draft of industrial systems and in different sequences during basic engineering of complex production systems (Mehr and Lüder, 2019). Based on the resulting system architecture, detailed engineering disciplines like lean engineering (Mrugalska and Wyrwicka, 2017), runtime simulations, electrical or mechanical engineering, and virtual commissioning (Lopez et al., 2022) could be executed with other tools focusing on this particular aspect. Thus, more practical applications ensuring the utilization of MBSE for designing the basic system architecture and providing it to detailed engineering disciplines are needed to foster its usability.

The paragraphs above imply that it is tough for manufacturers to adjust towards the IIoT. As Industry 4.0 is defined as a disruptive revolution (Koh et al., 2019), the need to adapt to this trend becomes apparent. In more detail, in the sense of product, process, and resource PPR systems, intelligent products are individually produced in lot size 1, production processes are flexibly accommodated, and production resources are digitalized. Moreover, the mentioned aspects are merging, and their dynamic interplay needs to be ensured. While there are already proposed methodologies to achieve such a stepwise transformation, their applicability for manufacturers is hardly investigated and should eventually be established (Calà et al., 2017). Following this argumentation, this thesis aims to combine the theoretical aspects of RAMI 4.0 as well as the practical concepts of MBSE to enable sustainable engineering of current and future industrial systems with a holistic and applicable approach. Sustainable engineering of new systems and the migration of already existing systems in the context of the fourth industrial revolution should be supported. This leads to the specification of the following main research question to be answered in the context of this thesis:

"How can domain-specific particularities and model-based engineering concepts be consolidated to a comprehensive modeling approach enabling holistic and interdisciplinary engineering of current and future manufacturing systems by addressing all stakeholder concerns?"

The proposed solution should allow and support manufacturers to adapt their traditional manufacturing lines towards flexible and reconfigurable production systems or design novel PPR systems in the context of the Industry 4.0 paradigm. To do so, the respective research artifacts are consolidated within the RAMI Toolbox, which provides a single framework to all practitioners. The main focus of this solution is feasibility, usability, and availability, aiming to reduce the barrier to entry for those manufacturing companies.

However, engineering such systems or transforming them along the aspects of the fourth industrial revolution is far-reaching and inherits a lot of different tasks to be considered. As stated in Calà (2019), the migration in terms of the fourth industrial revolution requires changes throughout the whole system life cycle and addresses technology, infrastructure, skills, and business models. Due to these statements, in the scope of this thesis, different aspects of the proposed systems engineering process are investigated and outlined in the remainder of this work. These aspects are considered to be the most promising needing to be achieved by manufacturers when defining a holistic industrial systems engineering approach.

The initial research has been guided by the proposal of RAMI 4.0. As this reference architecture model has been derived from the well-known and widely used Smart

Grid Architecture Model (SGAM), a distinction of domain-specific peculiarities between Smart Grid and Industry 4.0 is made. While MBSE is already established in the Smart Grid area (Neureiter et al., 2016), a feasibility study for transferring successful concepts is primarily conducted. This results in introducing a domain-specific language especially targeting Industry 4.0-specific characteristics. To achieve this, the successful approaches of SGAM are analyzed, and possibilities for applications in the industrial domain are investigated. This step results in a domain-specific metamodel and a process model for Industry 4.0.

A comprehensive modeling framework is proposed in the next step to enable holistic systems engineering of complex industrial systems and allow manufacturers to be equipped with a proper toolset. This framework falls back on well-known standards and technologies, which increases its acceptance within the community. For example, the theoretical foundations of RAMI 4.0 are utilized as a basis and extended by additional modeling activities. Integrating lean and agile methodologies ensures the application of MBSE within the modeling framework. This secures a holistic view of the system to be engineered and enables the continuous evolution of the whole system or parts.

Finally, as the proposed approach mainly focuses on the basic engineering of industrial systems, only an excerpt of the whole engineering toolchain is addressed. The last step of this thesis thus deals with integrating the modeling framework into the complete engineering life cycle. A toolchain is set up with the modeling framework dealing as a managerial tool to coordinate the remaining phases. For example, external business-related aspects are imported, while the resulting system is exported for further processing or detailed engineering. This allows us to consider the industrial system during its design, construction, implementation, commissioning, and operation. In the context of this thesis, suitable information exchange strategies and bi-directional interfaces are elaborated.

Different evaluation strategies are used to evaluate the developed concepts of the approach. In more detail, the Software-Architecture Analysis Method (SAAM) deals with assessing the proposed architecture of the modeling framework with a particular focus on feasibility, usability, and composability. This is done with the help of pertinent case studies, each of them evaluating another of the previously mentioned aspects. Thus, the domain-specific elements are reviewed using a fictive subway track manufacturer; the modeling framework is evaluated with the Siemens Fischertechnik model, and the toolchain is validated by applying a packaging station case study.

1.4 Delimitation

As far as the proposed framework is concerned, several delimitations must be determined. For example, the work presented in (Unverdorben, 2021) tries to achieve similar results by providing an architecture framework that derives system architectures based on reference architecture within the manufacturing domain. The RAMI Toolbox, proposed in this work, also represents a framework that allows users to engineer a system based on a reference architecture. However, as far as the RAMI Toolbox is concerned, an already established standard is used as a reference architecture, namely RAMI 4.0. However, Unverdorben (2021) additionally allows primarily defining and creating any reference architecture before deriving existing system architectures. This means a lot of effort must be put into creating reference architecture. On the other hand, in this particular thesis, this preparatory work has already been done by introducing RAMI 4.0. Concluding from this, this means that the main achievement of this work is to enable the application of the theoretical concepts originating from RAMI 4.0 and allow any interested practitioner to model system architectures based on the theoretical aspects. The underlying reference architecture is thereby provided by the RAMI Toolbox itself and its implemented features, like a domain-specific Language (DSL) or a metamodel.

Another necessary delimitation is the intended operation area of the modeling framework. The RAMI Toolbox mainly targets the engineering tasks that need to be achieved during the basic engineering phase: the development of abstract system architectures without technical details. During this phase, it is essential to interconnect all the system parts and define the main modules of the architecture. Implementation details or detailed elaborations are part of the detailed engineering phase, which is not addressed in the scope of this work. The resulting artifact of the modeling framework, a utilizable system architecture, is passed to the detailed engineering tasks. Those tasks are already established and executed with suitable tools. Therefore, competing with these successful approaches is pointless, while the barely defined basic engineering needs additional support. The RAMI Toolbox only offers essential functionalities to create an abstract system representation. Figure 1.3 gives a better overview of the intended operation area of the proposed framework and its classification within the engineering process according to the VDI 2206 guideline (Gausemeier and Moehringer, 2002).

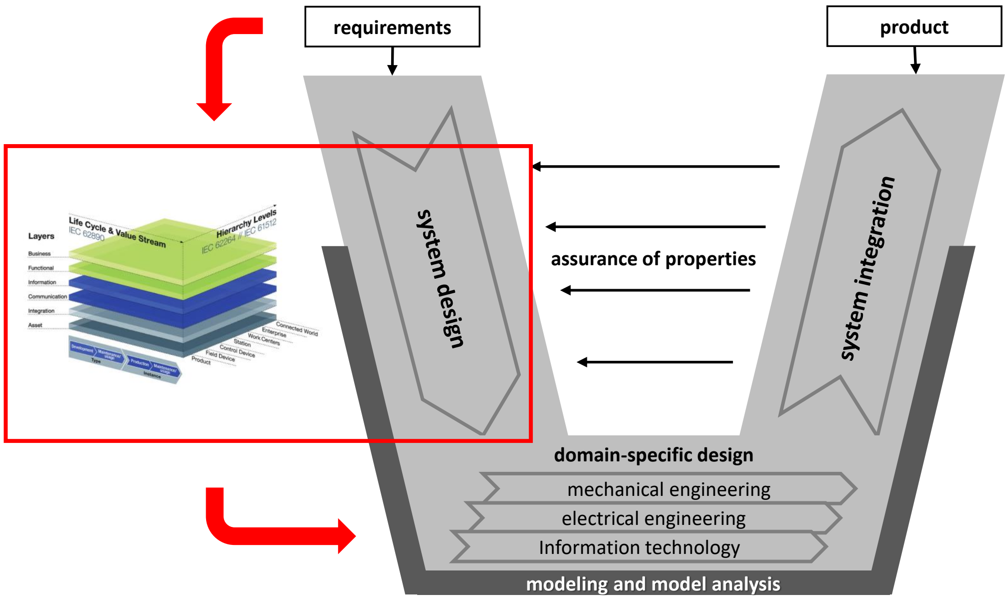


Figure 1.3: Classification of the RAMI Toolbox by the VDI 2206 engineering process

It is shown that the RAMI Toolbox is mainly used for describing the system design, which is for basic mechanical systems engineering. A domain-specific design like mechanical, electrical, or information technology engineering is not part of this work and should be done based on the previously created system architecture description. This also counts for finally integrating the system. Thus, the primary purpose of the modeling framework is to interconnect all system stakeholders and regard their concerns as requirements for the system to be developed. Those requirements are considered and fulfilled during the basic engineering phase while creating the main system design as architecture. Finally, this architecture is passed to detailed engineering phases via suitable interfaces and data formats to enable more specific engineering. Thus, only representative examples are used for addressing subsequent tasks of the engineering toolchain, as considering all possibilities would exceed the scope of this work. This means all elaborated artifacts of this work are targeted to detailed engineering, as enclosed by the red frame within Figure 1.3.

1.5 Thesis Outline

To further explain the planned outcome and conducted research of the industrial systems engineering approach, this thesis is structured as follows: Chapter 2 deals with giving an overview of currently used methodologies or techniques in the area of industrial systems engineering and provides the foundation for the present work. The purpose of the following Chapter 3 is to explain the related work and fundamental background, particularly with a focus on architecture development and systems engineering. Subsequently, Chapter 4 outlines the applied research methodology and introduces the case studies used to evaluate the research results. The central part of this thesis is done in the following chapters, which respectively summarize the implementation and application of the DSL in Chapter 5, the modeling framework in Chapter 6 and the toolchain integration in Chapter 7. In the following, the outcome is evaluated based on the case studies in Chapter 8, where the findings are also located. Finally, Chapter 9 summarizes the conducted research and provides an outlook into follow-up projects. Figure 1.4 outlines the described paragraph and shows the logical chain of this work and the interrelations between the chapters considered in this thesis's context.

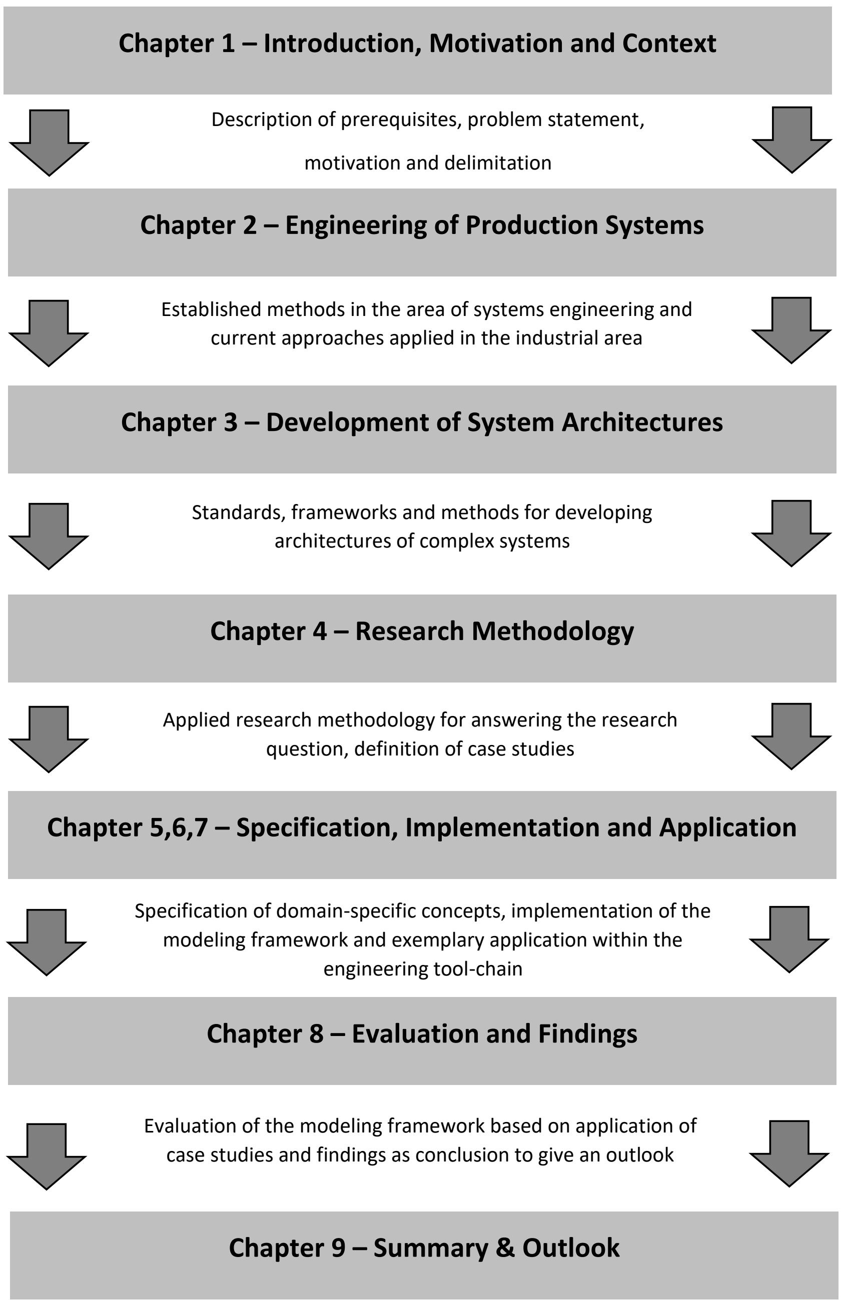

Chapter 1 – Introduction, Motivation and Context
Description of prerequisites, problem statement, motivation and delimitation
Chapter 2 – Engineering of Production Systems
Established methods in the area of systems engineering and current approaches applied in the industrial area
Chapter 3 – Development of System Architectures
Standards, frameworks and methods for developing architectures of complex systems
Chapter 4 – Research Methodology
Applied research methodology for answering the research question, definition of case studies
Chapter 5,6,7 – Specification, Implementation and Application
Specification of domain-specific concepts, implementation of the modeling framework and exemplary application within the engineering tool-chain
Chapter 8 – Evaluation and Findings
Evaluation of the modeling framework based on application of case studies and findings as conclusion to give an outlook
Chapter 9 – Summary & Outlook

Figure 1.4: Thesis outline

Chapter 2

Systems Engineering in the context of Industry 4.0

This chapter gives an overview of the theoretical background in the context of industrial systems engineering and currently used approaches. Thereby, generally established methodologies are introduced, as well as particular ones that mainly target this domain. The respective utilization within the RAMI Toolbox and an eventual delimitation are also considered.

2.1 Systems Engineering

The definitions of a system are as broad as the breadth and depth of terms it covers. On the one hand, systems are as far-reaching as the universe itself. On the other extreme, they are so detailed that even the smallest atoms can be imaged. Today there are various systems that humans have created (Blanchard and Fabrycky, 2011). At first glance, this comparison seems disproportionate, but it gives an overview of the areas of responsibility of system engineering. In the classic sense, it expresses the endeavor to bring all experts in different disciplines together and bring them to the same level of information. As a result, individual problems that come together in a large complex can be dealt with by a specialist in the respective field.

According to Lightsey (2001), engineering such systems consists of two main disciplines. On the one hand, it is necessary to generate technical knowledge about the system's domain. On the other hand, system engineering management contributes significantly to the success of the result. Such an approach aims to determine the best solution with all requirements, a timetable, and consideration of costs. Especially in the economic area, it is vital to ensure that the essential components of the corporate strategy are included. The trend towards systems with a higher level of information exchange concludes that traditional system engineering has to be reconsidered. The result, shown in Figure 2.1, is an extension of the original method, including new responsibilities, named enterprise systems engineering (ESE) (Carlock and Fenton, 2001).

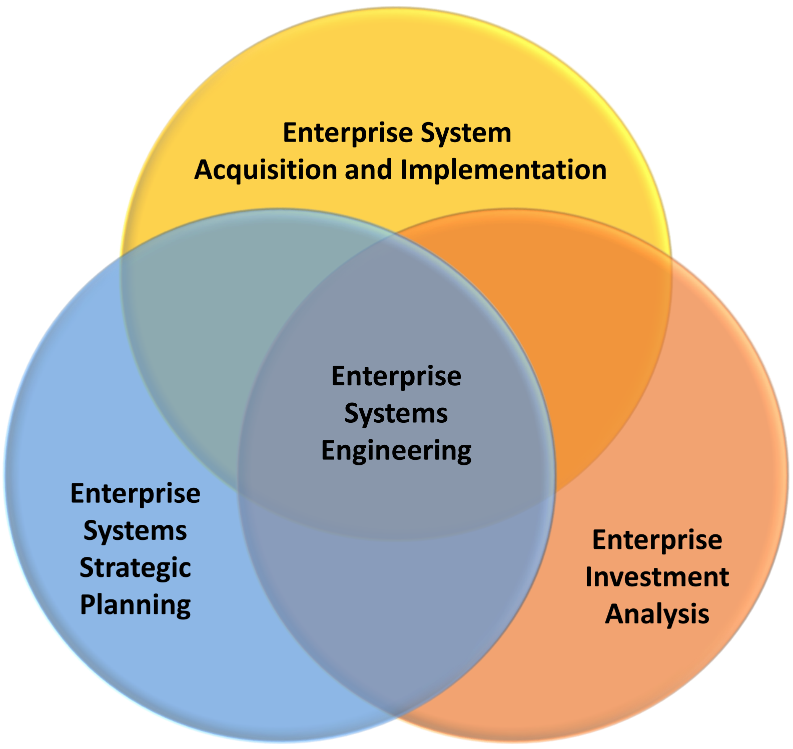

Enterprise System Acquisition and Implementation
Enterprise Systems Engineering
Enterprise Systems Strategic Planning
Enterprise Investment Analysis

Figure 2.1: Enterprise system engineering based on (Carlock and Fenton, 2001)

This approach originates from methods of traditional business management. It intersects strategic planning, investment analysis, and acquisition and implementation. A unique feature is that no single system is constructed in this case. The interaction of several homogeneous systems, a system-of-systems (SoS), allows the continuous recording of company-wide processes without having to sacrifice individual production potential (Carlock and Fenton, 2001). Putting the concept of SoS into reality requires different methods than those used in developing single systems, as various characteristics are defined differently. The main difference between systems and SoS originates primarily from their functionality. While a single system represents a solution for a specific purpose, their combination provides a concept for a more complex use case. However, it should be noted here that a SoS is not the sum of the resources of the respective subsystems. The following paragraph outlines the difference between a system and a system of systems. Boardman and Sauser (2006) compiled and analyzed them, ultimately proposing five terms that essentially describe the approach of SoS.

Independence and Diversity

Systems exist to pursue a certain goal. To achieve this goal, they inherit a degree of freedom limited by conditions. In doing so, the operation of the system must not be impaired. Otherwise, the necessity for its existence must be questioned. This is different for SoS. A single system does not work for its own goal but subordinates itself to the whole. Therefore, they are usually designed in such a way that they serve this one purpose. The adaptation thus limits the autonomy of such a subsystem to the overall system. For this reason, independence must be questioned. Even if the definition of a system applies to it, it is treated more like a part. (Boardman and Sauser, 2006)

A rule by Ashby (1991) states that a system must have the same degree of freedom as its dimensionality. On the other hand, an attempt is made to limit the functionality to the essentials. By linking these two approaches, the complexity of a large system can be mastered. Consequently, a SoS should integrate this behavior to show increased diversity compared to a normal system. The individual parts concentrate on accomplishing the tasks intended for them, which ensures abstraction. (Boardman and Sauser, 2006)

Affiliation, Connectivity, and Appearance

Former autonomous systems becoming part of a larger complex have little experience in integration. To work together with other units, relationships must be formed. Core issues of value and autonomy should be addressed when creating a SoS. To ensure collaboration, an appropriate infrastructure should be set up. The resulting connections between the units are encapsulated to appear hidden within the system and are only carried to the outside via interfaces. This facilitates management and administration. It also allows for dynamic interoperability in which units can be easily swapped out.

Similar behavior can be seen in the design. In systems, appearance is consciously considered from the start. This is hardly possible with a SoS since there is too much uncertainty due to the complexity and dynamics. The design is, therefore, an ongoing process and must be constantly adapted, leading to risks. (Boardman and Sauser, 2006)

As the term of SoS has been discussed before this millennium, several attempts to categorize such systems have been made. Thus, it was clear from the beginning that current and future industrial systems will inherit the characteristics represented by these systems. To give further details, the early classifications and typical characteristics of such systems have been proposed by Maier (1998). In his publication, he first explains his point of view about SoS, which is subsequently underpinned by several characteristics. Thus, he claims that a SoS is not descriptive in a formal sense. The term implies a taxonomic grouping of system parts, representing distinct classes representing design, development, and operation demands. A single system is an assemblage of components producing a behavior or function not available by any component individually. A SoS assembles components regarded as systems. Additionally, the complexity of such systems also play an important role regarding them as systems themselves. Thus, Maier (1998) derived two additional characteristics for defining a SoS:

1. Operational Independence of the Components: If the SoS is disassembled into its component systems, the component systems must be able to operate independently usefully. That is, the components fulfill customer-operator purposes on their own.
2. Managerial Independence of the Components: The component systems not only can operate independently, but they do also operate independently. The component systems are separately acquired and integrated but maintain a continuing operational existence independent of the SoS.

DeLaurentis (2005) apprehended this definition and additionally explained that categorically defining the SoS problem is counterproductive due to focusing on the artifact rather than on the actual challenge. Thus, by not defining the term but rather recognizing traits of the problem type, a categorization of SoS is done, as the development of effective methods will depend on this understanding. Thus, the elaborated characteristics are summarized and introduced in Table 2.1.

As identifiable from the previous paragraphs, contemporary and future industrial systems are composed of several subsystems to form so-called SoS. As a result, these

Table 2.1: System of systems characteristics by DeLaurentis (2005) and Maier (1998)

Characteristic	Description
Operational & Managerial Independence	Constituent systems are useful in their own right and generally operate independently of other systems.
Geographic Distribution	Constituent systems are not physically co-located but in communication.
Evolutionary Behavior	The SoS is never completely, finally formed; it constantly changes and has a “porous” problem boundary; this means a SoS is a living system.
Emergent Behavior	Properties appear in the SoS that are not apparent as well as predicted from the constituent systems.
Networks	Networks define the connectivity between independent systems in the SoS through rules of interaction.
Heterogeneity	Constituent systems are significantly different, with different elementary dynamics that operate on different time scales.
Trans-domain	Effective study of SoS requires unifying knowledge across engineering, economics, policy, and operations.

systems are becoming increasingly modular, considering the interests of many stakeholders and allowing greater freedom in decision-making. As a result, such industrial systems’ creation, analysis, management, and adaptation are constantly made more difficult. Another consequence of this increasing trend is that conventional systems engineering methods are no longer sufficient to master the complexity of the SoS. For this reason, model-based systems engineering is considered one of the most promising approaches to deal with this increasing complexity and offers a constant development process over the life cycle. It also simplifies administration and collaboration in such projects considerably.

The previous paragraphs underpin the importance of the following three sections, as they all provide techniques for elaborating parts of the system while simultaneously dealing with the complexity of SoS. MBSE mainly utilizes a single model for documenting the system rather than using different information sources, requirements engineering deals with consistently elaborating and specifying system requirements, and round-trip engineering (RTE) ensures consistency between model and implementation.

2.1.1 Model-based Systems Engineering (MBSE)

Wymore (2018) describes MBSE as a formalized application of modeling to support systems engineering activities during the design phase and construction of the system. This includes requirements engineering, architecture development, system analysis, and verification and validation. For this reason, a system model is created, representing an abstract reflection of the system to be developed. This model is created during the planning phase of the system and maintained after its implementation. It accompanies the systems engineering from the idea to the solution of the system (Weilkiens, 2011). The model should have the following properties (Weilkiens et al., 2015):

- The entire model can consist of different databases, but these must be consistent and should appear to the user as a single model.
- The system model must allow several different points of view.
- The system model must be computationally evaluable and provide an abstract syntax. This should include and support explicit MBSE concepts such as requirements engineering.

There are two widely used approaches to describe the architecture of a system. On the one hand, the traditional document-based approach is often used to describe software and systems. Here, key artifacts that affect the architecture are stored in specific documents, and the information is drawn from them if necessary. On the other side, in the MBSE, the data is stored centrally in a model that counts as an access point for all information. Changing requirements or the system design can be costly and resource-intensive with the document-based approach. The distributed storage of information can also result in inconsistencies. In this respect, the model-based approach is the better solution because the model is shared across all stakeholders, and changes affect all related elements. In addition, the previously outlined design principles are addressed by MBSE.

To apply MBSE in actual projects in the field of software or systems engineering, various model-based approaches are provided (Brambilla et al., 2017). As shown in Figure 2.2, these are subdivided into model-based engineering (MBE), model-driven engineering (MDE), model-driven development (MDD) and model-driven architecture (MDA). The graphic shows the relationship between the individual approaches, which are defined as follows:

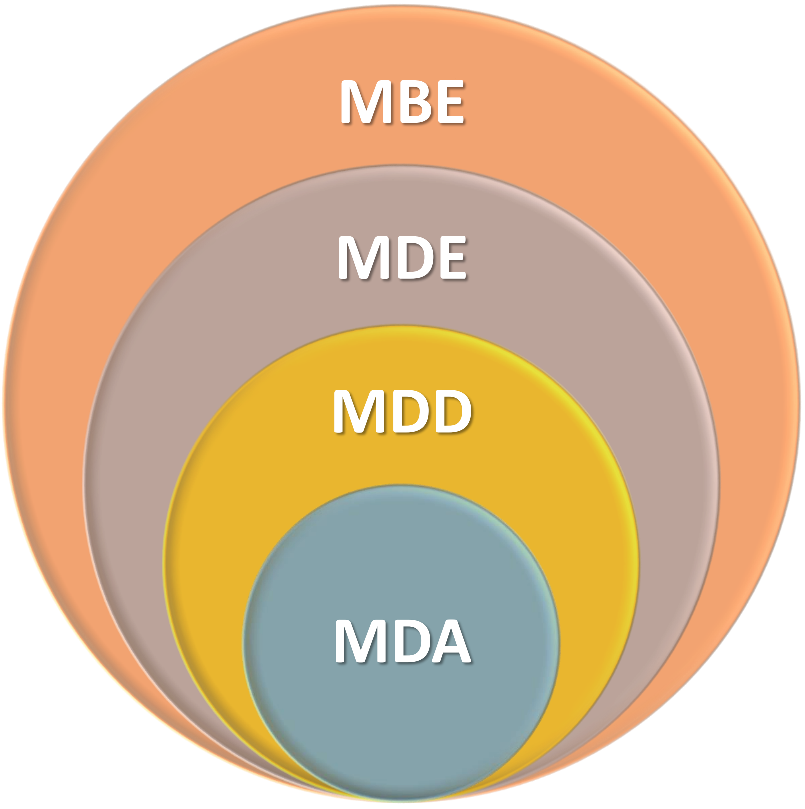


Figure 2.2: Model-based approaches based on Brambilla et al. (2017)

Model Driven Architecture (MDA)

MDA is Object Management Group (OMG)'s (Place, 2000) specific view of how MDD should be applied. This type refers to OMG standards that unify modeling and transformation languages. As a result, MDA is a widespread and standardized approach to developing complex systems. The four principles that underline the utilization of MDA explain that well-defined notations are the cornerstones for understanding systems, which might be organized around a set of models and by imposing a series of transformations between them. Additionally, a metamodel should facilitate the integration and transformation of models, which also need to find acceptance within the community. (Brown, 2004)

Model Driven Development (MDD)

MDD has been defined with the general goal of using models as the primary artifacts of the system development process. The implementation is ensured by the system being generated semi-automatically or entirely automatically from the models. MDD thereby defines the utilization of models to design complex systems in traditional engineering disciplines (Selic, 2003).

Model Driven Engineering (MDE)

MDE is superior to MDD because it is assumed that manual or automated engineering activities go beyond the usual development steps. This also includes other tasks of the complete system engineering process, which can be model-based evolution or reverse engineering of already existing systems. However, MDE technologies offer promising methods to address the complexity of platforms and express domain concepts effectively (Schmidt, 2006).

Model Based Engineering (MBE)

This type of systems engineering refers to a more superficial version of the model-based approaches. Thereby, models play an essential role during the engineering process but do not necessarily have to be the key artifacts. Examples of this type would be design models, which are used as blueprints for software developers, but they write the code manually. The model plays a vital role in this way but is not the central aspect of the system engineering process. MBE mainly deals with identifying opportunities to improve systems engineering productivity and completeness (Bergenthal, 2011).

Systems Engineering Management

Management is crucial to the quality of the end product. Three distinct roles help in the structured management of the entire system-building process. In the development phase, the first concept is created. In addition, requirements and components are described in detail. In the Life Cycle Integration, the finished design is replaced by a design solution that meets the requirements previously developed. A system engineering process runs parallel to these two phases. This exists throughout the development phase and is used to transform requirements into actual specific solutions. Information is provided, and the transitions between the individual stages are managed. The transitions between the individual task areas, shown in Figure 2.3, are realized by generating a common basis, cooperation, and planning between the people involved. (Lightsey, 2001)

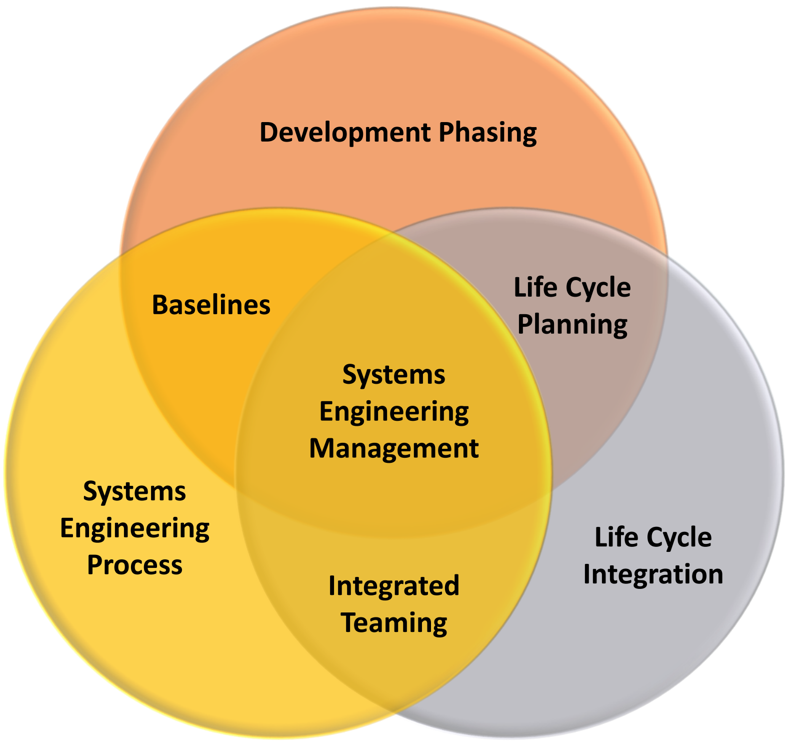


Figure 2.3: Systems engineering management based on (Lightsey, 2001)

2.1.2 Model-Based Round-Trip Engineering

The concept of MBSE defines a model to be the foundation of the overall systems engineering process (A. L. Ramos et al., 2011). Therefore, multiple systems engineering activities can be supported, including detailed requirements definition, validation, and verification. Moreover, the flawlessness and quality of system design are promoted. Risks in requirements definition, systems design, integration, and testing can also be reduced. Another part of MBSE is simulations, mainly of behavioral diagrams, to verify the model's completeness and flawlessness. This can be achieved through automatic code generation based on behavioral diagrams. The resulting software can then be executed on target hardware. (Borky and Bradley, 2018)

An essential aspect of such model-to-code transformations is bi-directionality, to keep the code consistent with the model and vice versa. On the one hand, if the model is changed, the code generated prior should adapt accordingly. On the other hand, if the generated code is changed, the model is also expected to adapt. However, this process would require a permanent, bidirectional connection, which is somewhat complex and often not worth realizing. A different, more feasible approach is RTE. In RTE, this connection is not maintained; it is realized by creating the code from the model and the other way around. (A. Kleppe, 2008)

The RTE process is used for any model-to-model transformation, which thus also counts for model-to-code generations. Therefore, the goals in model transformations are the same as in code generations. The source and target need to be consistent. Due to maintenance or altered requirements, changes to one model should also be adapted to the other model. A challenge of Model RTE was outlined by Hettel et al. (2008): Transformations between a source and a target model can not always be executed one to one, as either the source or the target model may include aspects that cannot be mapped to the other. Hence, an attempt has been made to find a formal definition for partial and non-injective transformations in RTE. Overall, their research concluded that the outlined challenge in Model RTE is highly complex since, before the transformation process, the semantics of model changes must be defined. Therefore, changes between models might have to be restricted in scale.

An example of a successfully implemented process model enabling RTE for adopting and evolving production systems has been proposed in (Krüger et al., 2020). The introduced "promote-pl" framework offers different process model elements and adaptions for different stages of the RTE process, like product-line management, product-line adoption, product-line evolution, and domain and application engineering. This process model's main advantage is allowing practitioners to quickly map and even apply RTE activities to production system development processes, as the authors claim. However, while most other approaches deal with RTE between the system model and its implementation, the method presented in this work explicitly targets the interconnection between basic and detailed engineering.

2.1.3 Requirements Engineering

Requirements engineering mainly concerns the identification of goals, which have to be achieved by the envisioned system (Van Lamsweerde, 2000). Those goals might then be operationalized into services and constraints and assigned to agents, such as humans, devices, and software, who take over responsibilities. Various processes are utilized, which consider single steps such as domain analysis, elicitation, specification, assessment, negotiation, documentation, and evolution. Reasonable requirements engineering could thereby result in better modeling of the problem domain or richer models for capturing and analyzing non-functional requirements, according to Nuseibeh and Easterbrook (2000). In addition, requirements engineering bridges the gap between the elicitation approaches and the formal specification techniques.

Successful systems engineering needs to be based on a definition of system requirements that is complete and free from errors. Especially the rising significance of systems with a substantial proportion of software requires efficient requirements engineering to ensure this. Requirements engineering describes the process of defining the critical requirements of the system of interest (SoI) (Sommerville and Sawyer, 1997). A key term of this process is the stakeholder. This term describes either a person or an entity represented by a person who affects the requirements of the SoI directly or indirectly. The stakeholder needs are evaluated, documented, and verified during the requirements engineering process according to quality criteria. The resulting documentation should facilitate communication between stakeholders and improve the quality of established requirements. Most commonly, natural language is used as a form of documentation. An advantage of this approach is that the stakeholder is not forced to acquire a new form of notation. However, by using natural language, there is also a danger that requirements are ambiguous so that information about specific requirements may become intermingled (Pohl and Rupp, 2015).

An alternative for language-based documentation is a graphical approach. For instance, requirements diagrams based on Systems Modeling Language (SysML) can be utilized to realize model-based requirements documentation. A benefit of this kind of documentation is that dependencies between requirements and other elements of the system can be visualized. System elements are also needed for an appropriate realization of the identified requirements. Typically those elements are added to the model alongside the relationships between these new elements and the respective requirements. Hence, traceability of the relationships between requirements and other system elements can be obtained (Delligatti, 2013). Despite the abovementioned benefits, documentation realized by modeling a requirements diagram is not truly model-based. As outlined in (Salado and Wach, 2019), a requirement displayed in a SysML diagram depicts its textual definition. However, the requirements are not captured appropriately. Instead, a theoretical framework is proposed in which requirements can be modeled as input/output transformations. The requirement, in this case, is considered as a system itself, while interfaces realize the respective in- and outputs.

Generally speaking, it can be stated that model-based requirements engineering is considered to be an essential factor when it comes to describing current or future systems. This is underpinned by numerous approaches introducing promising concepts. However, a big issue with the mentioned contributions is their generality. Being targeted to be applied in many systems spanning various domains, only superficial aspects can be considered. Thus, with the industrial environment, although being an area relying on well-defined requirements, the proposed approach builds on established requirements engineering methods. This will not only enhance the quality of RAMI 4.0 based models but also lead to traceability between model and implementation and a more comprehensive systems engineering.

2.2 Architecture Development

“If all you have is a hammer, everything looks like a nail“, has been phrased by the American psychologist Abraham Maslow. This quote can be interpreted in different life situations, as also in the area of systems engineering. It states that a successfully created solution does not have to have the same positive effect on the next project. This applies to the industrial sector because this domain is exposed to constant change. Due to new technologies, changes in conditions, or advances in the value-added process, the system must be constantly adapted. The descriptive software architecture of this system is not excluded from this adjustment, which concludes that value should be placed on a dynamic structure using up-to-date methods. (Vogel et al., 2008)

Within this context, it is essential to distinguish between software and production systems architecture (Unverdorben, 2021). While software architecture is fundamental in software-intensive systems (Rozanski and Woods, 2011), architectures of complex production systems or reference architectures gain importance to depict the many modules and wide-ranging interrelationships of the system. Thereby, systems architectures fall back on the main principles of the early days when describing software architectures. However, to cope with the complexity, new approaches needed to emerge. Thus, the following principles of developing software architectures are outlined, followed by an overview of approaches to creating system architectures.

The terminology of the IEEE standard 1471-2000 for describing software architecture (Hilliard, 2000) defines software architecture as the primary organization of a system, which is represented by its components, their relationships to one another, and principles that determine the design and evolution of the system. Each kind of software has its architecture. However, this is not always visible at first sight. In addition to technical aspects, software architecture includes social and organizational factors, making it an essential part of Industry 4.0 due to its complexity and diversity. Among other things, no details of the architecture are addressed, mainly the framework and its pillars are defined. (Vogel et al., 2008)

2.2.1 Terms and Concepts

Several principles are specified in modern software architecture. These influencing factors should be observed as far as possible so that the result meets the quality requirements. The most important principles are explained below:

Principle of Loose Coupling

Modules, components, classes, and other building blocks of a system are related, referred to as coupling. This ensures the interaction between the individual elements and is influenced by the type of communication and data exchange. The principle of loose coupling means that the complexity of the structure between the building blocks should be kept as low as possible. This is done by making a building block self-explanatory and having direct dependencies on other building blocks, such as global definitions, which also increases the changeability. The definition and realization of interfaces are determined by the principle of dependency inversion (Martin, 1996). This states that elements provide interfaces to control access. (Vogel et al., 2008)

Principle of High Cohesion

While coupling deals with the dependencies between the building blocks, cohesion describes dependencies within the building block. This also expresses local changeability and understanding of the functionality of the block. The cohesion should be as high as possible so that the building block can be changed without affecting other building blocks. Thus, coupling and cohesion are interrelated; they influence each other. (Vogel et al., 2008)

Principle of Design for Change

This principle describes the constant change of software. The associated architecture should be developed so that expected changes are quickly taken into account and anticipate unforeseeable changes with certain flexibility. However, this entails higher implementation and resource costs. However, considering the principles of loose coupling and high cohesion, this principle can be decisively influenced. (Vogel et al., 2008)

Separation-of-Concerns-Principle

Separation of concerns defines the separation of the areas of responsibility and the individual treatment of sub-problems. This results in the system being broken down into understandable and coherent individual parts, which often happens due to functional requirements. Doing so allows functional and technical components to be separated from one another. Other principles, such as abstraction or modularity, are based on this principle, as they also try to break down a complex issue into manageable and suitable parts. (Vogel et al., 2008)

Divide-and-Conquer-Principle

With this principle, the complex issue is broken down into smaller, manageable subunits. These units are called procedures, each representing a logical unit that solves specific tasks. These subprograms are called up by the program above them, which does not receive any information about the execution. Thus, the implementation does not have to be known; only the interfaces have to be defined. (Bentley, 1980)

2.2.2 Model Driven Architecture (MDA)

MDA is an approach to the visualization, storage, and exchange of software designs and models. To this end, it builds on well-known industry standards such as Unified Modeling Language (UML). This allows qualitative models to describe long-lived systems and multiple technologies to implement them (A. G. Kleppe et al., 2003). MDA attempts to subdivide an architecture into different levels of abstraction and automatically merge them into one another. This separates the information logic from the technical platform. A framework is provided for this purpose, which defines the framework conditions. MDA divides the SoI into models and transformations. Models describe the system, with a single model describing a specific part of the whole. Transformations describe the automatic transfer of a model to another level of abstraction. In addition, the developer can decide which aspects of the source model should be transferred to the target model (Posch et al., 2012). Figure 2.4 represents the principles of MDA, including models and transformations.

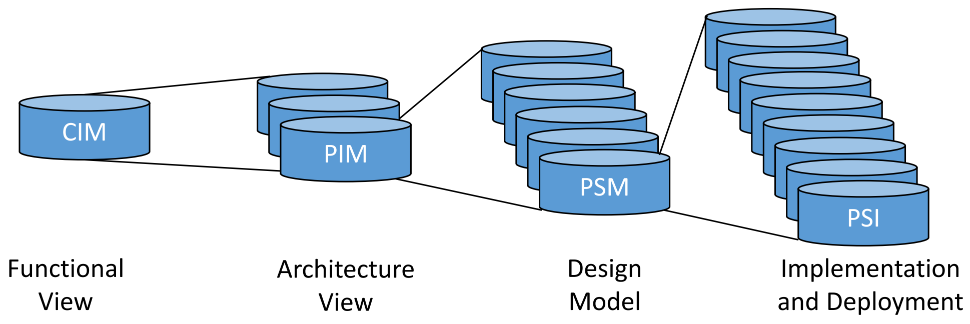


Figure 2.4: Model Driven Architecture based on Watson (2004)

MDA attempts to model the specific implementation and associated code. A platform-independent model can be realized automatically through mutually coordinated specifications and levels of abstraction. The system is administered via the models and associated transformations, which are described below:

Computation Independent Model (CIM)

The Computational Independent Model (CIM) describes a system in a way that end users can understand. The requirements are included, and the functionality of the result is clarified. A generally understandable language is used to make the scope of services understandable for everyone involved. Specific technologies are not part of this model. A transformation transfers the CIM findings to the Platform Independent Model (PIM). (Kempa and Mann, 2005)

Platform Independent Model (PIM)

Components in this model represent the previously specified system properties. These are writable and have certain functionality but are classified independently of the platform. The focus is, therefore, on the utilization of the respective components. In MDA, a platform is a technology that can be used over an interface. In more detail, services are thus provided, but the exact implementation might be unknown. Subsequently, a transformation to the more specific Platform Specific Model (PSM) adds further details. (Kempa and Mann, 2005)

Platform Specific Model (PSM)

In this model, the components from the PIM are divided into platform-specific units, mainly done using the provided interfaces. For a detailed technical description, the integration of details from the selected platform is carried out so that the infrastructure for the automatic generation of the program code can be developed. (Kempa and Mann, 2005)

Platform Specific Implementation(PSI)

In the last model, the Platform Specific Implementation (PSI), the program code, is represented and managed. Thus, the technologies used are provided with an exact level of detail. The code can be generated manually, semi-automatically, or fully automatically from the CIM, PIM, and PSM specifications by executing the transformations. These models should be combined with and used later for the detailed disciplines of production systems engineering within various domains. (Kempa and Mann, 2005)

2.2.3 Metamodeling

As the name assumes, metamodels are models that describe the modeling language. A metamodel defines the structure and meaning of specific models. As a rule, a simple metamodel is developed at the start of a project as a base for the future system. Findings and experiences during the creation of the system are then adopted, which means that the metamodel is constantly adapted. Various syntaxes can be used to describe a metamodel. The concrete syntax can be either graphical or textual, explaining the same concept. Conditions represent the abstract syntax, so the metamodel is considered valid. (Stahl and Volter, 2006)

There are different ways of designing metamodels. The ongoing adaptation and revision of the model to the constantly changing requirements is described under a lively metamodel. Viewing the metamodel as the primary architecture driver is another possibility. This means prioritization and preferential treatment compared to templates and transformations. Creating a separate metamodel for several aspects of the system is another option for separating and defining responsibilities. (Stahl and Volter, 2006)

With CPSs, it is important to analyze the physical world and determine its properties. From this, the behavior of the physical domain can be analyzed and thus create a model of the actual world. Finding the connections between the physical and virtual worlds is important to generate a metamodel. The abstractions of the metamodel are defined from the semantics and structure of the physical model. (Mezhuyev and Samet, 2013)

In recent years, multiple metamodels have been proposed, mainly focusing on Industry 4.0. One of the first research projects introduced a metamodel for integrating Industry 4.0 with the IoT, social networks, and the cloud (Molano et al., 2018). The core aspects of each mentioned domain are considered by falling back on foundation principles, available elements, and suitable technologies. The resulting metamodel should thus support the generation of applications for Industry 4.0, which is investigated with a case study. More recently, enterprise readiness for Industry 4.0 has been investigated (Basl and Doucek, 2019). By also considering RAMI 4.0, multiple other reference architectures have been collected and summarized. The resulting proposal for a metamodel consists of various levels, each giving a more composed view of a particular aspect. Other considerations could be made on the lower levels, Level 5 to Level 7. As those levels consider aspects in a specific enterprise, the metamodel more or less falls back on the criteria of RAMI 4.0 at this granularity.

However, as big data is becoming increasingly critical in the context of Industry 4.0, a data-centric architecture metamodel has been proposed by Martıénez et al. (2021). The authors claim that the characterization of managed data at the highest level of abstraction defines the industrial environment. To produce, process, or consume this global data, a data-centric industrial platform builds the starting point for the reference architecture. All hardware resources that comprise this platform thereby provide data as a service. On a lower granularity level, the platform as a service deals with the resources needed to handle lower-level data. Those resources comprise embedded systems, components dealing with daily operations, and cloud-operated components, which allow the instantiation of distributed, heterogeneous, and scalable platforms. Finally, the proposed metamodel defines monitoring as a service on the bottom level, which deals with horizontal and vertical scaling.

2.2.4 Domain-specific Languages (DSLs)

Software architectures and models often work in a specific area with changing conditions. A metamodel can abstractly describe the structure of this area. Hence, a DSL can play a decisive role in the description and development of this metamodel in that the development is steered in a strong direction through the definition of a concrete syntax and adequate semantics. This syntax and semantics should be selected so the user can get along with the domain-specific terms and ensure a working environment. (Stahl and Volter, 2006)

A DSL is thus a declarative language that focuses on the problems of a domain. It offers suitable notations and abstractions. They often contain another sublanguage, a so-called general purpose language (GPL), which gives them additional expression. In this way, DSLs promotes a system's productivity, portability, maintainability, and reliability, which is of considerable importance in the industrial sector. (Van Deursen et al., 2000)

There are three types of DSLs related to the application dependency. An external DSL is a separate language that acts as a support to the main language of the application. The definition of an internal DSL says that the application language is transformed into a domain-specific language. Thus, with an internal DSL, the system is based on this. This type is often confused with an embedded DSL. However, embedded languages are sub-languages of a larger system, such as Visual Basic for Applications (VBA) in Word. On the other hand, internal DSLs are independent and do not require a higher-level application. (Fowler, 2005)

2.2.5 Reference Architectures

The results of the principles and methods mentioned are usually so-called reference architectures. These combine general architectural knowledge with the specific requirements of a particular problem area to create an overall solution. In doing so, they define the essential components and the standard functionality of the future system, reflected in a reference model. This leads to a higher quality solution while reducing costs and saving time (Vogel et al., 2008). Although the term reference architecture is defined differently across the industry, it is described in (Cloutier et al., 2010). The authors explain that reference architectures combine the essential components of existing architectures with future requirements to assist in developing new system architectures. Different architecture drivers have emerged across all domains. These include the high complexity of systems and the increasing dynamics and interoperability in the industrial sector. Therefore, a reference architecture is strongly tied to the company's vision and strategy (Cloutier et al., 2010).

Today, DSLs are widely used within systems engineering, either to provide an ecosystem for threat modeling (Hacks and Katsikeas, 2021), enabling the possibility to model dependencies for adding value to design processes or contributing to consistency (Qamar et al., 2013) or to capture design synthesis knowledge aiming to compose well-defined components into larger systems (Kerzhner and Paredis, 2009). Thus, while this concept becomes increasingly popular in current systems engineering, Salman et al. (2021) investigated the challenges and opportunities of such a DSL within the IoT area. By analyzing more than 1000 papers, two groups have been recognized for the most recent efforts towards developing a DSL dedicated to the IoT. First, the architecture and development criteria are mentioned, which are the main factors when it comes to forward IoT technologies to end users. This allows them to develop more sophisticated applications without needing to cope with low-level technical details. Secondly, DSLs is applied to consider the development and evaluation criteria to aid end users during the early stages of systems engineering. As both approaches also inherit negative alongside their positive effects, further research needs to be done in this area (Salman et al., 2021).

2.3 Architecture Standards

2.3.1 IEEE 1471

The IEEE 1471 (Hilliard, 2000) documents good architectural description practices of the software. Thereby, conceptual issues and their associated difficulties are addressed during the standard development process. It has been developed as defining architecture in the computing context became one of the most contentious issues. While this standard mainly targets software architecture, besides software-intensive systems, more general systems, like information systems or SoS, could be addressed.

For describing architectures, IEEE 1471 focuses on description requirements in terms of its elements. The system should specify its stakeholders and identify their concerns within the architecture. The main questions to be considered in this particular example are the functionality of the system, in more detail, what should be achieved by it. The following questions deal with the system's performance under heavy loads and if it can protect user information in a good way in terms of security. Finally, the feasibility needs to be investigated if it's ultimately possible to implement the system. (Maier et al., 2001)

The foundation of the approved version of the IEEE 1471 is thereby built by five core concepts and relationships, which are (Maier et al., 2001):

1. Every system has an architecture, but architecture is not a system.
2. An architecture and an architecture description are not the same thing.
3. Architecture standards, descriptions, and development processes can differ and be developed separately.
4. Architecture descriptions are inherently multi-viewed.
5. Separating the concept of an object's view from its specification is an effective way to write architecture description standards.

In later publications, the gap between the IEEE 1471 and systems engineering has been reduced by additionally providing a architecture description language (ADL), namely UML. The approach instantiates the conceptual framework of the standard and enables the utilization of its theoretical abstractions and mechanisms. In more detail, viewpoints for structurally describing software architecture are provided, which support the key concepts of ADLs by utilizing an UML profile as viewpoint language. (Kandé et al., 2002)

Finally, as initially defined as a software-focused standard, its equal utilization for any given system has been argued and ultimately substantiated by Maier et al. (2004). The demonstration and applicability in systems engineering to describe system architectures are analyzed and fundamentalized. Based on the results, ISO 42010 has emerged from promising concepts.

2.3.2 ISO 42010

Standardized approaches are needed to counteract the increasing complexity of industrial systems. The ISO 42010 (International Organization for Standardization, 2011) represents such an approach. By doing so, the architecture development of complex systems is supported by its general concepts. In detail, methods for creating the system, its analysis, maintainability, and communication between the individual stakeholders are provided. The standard itself is shown in figure 2.5, where the left column is particularly important. This means that ISO 42010 can be used to describe industrial system architectures. In addition, it is possible to develop architecture frameworks and so-called ADLs, with which metamodels of the respective systems can be created. Thus, the core of ISO 42010 is an ontology, as Tschirner (2015) explains.

Thereby, the ISO 42010 inherits the concept of model correspondences, which are also provided in the 4+1 view model (Kruchten, 1995) as well as the Reference Model of Open Distributed Processing (RM-ODP) (Putman, 2001). However, model correspondences in the ISO 42010 are compatible with viewpoint correspondences within RM-ODP. According to Emery and Hilliard (2009), it has to be noted that viewpoint correspondences are binary, whereas model correspondences are n-ary. Additionally, while viewpoints are homogeneous, a single language that interconnects elements within the viewpoint might be used. In contrast, model correspondences could also interconnect models along modeling elements or have no reference. This means that ISO 42010 allows heterogeneous views composed of models with different modeling languages.

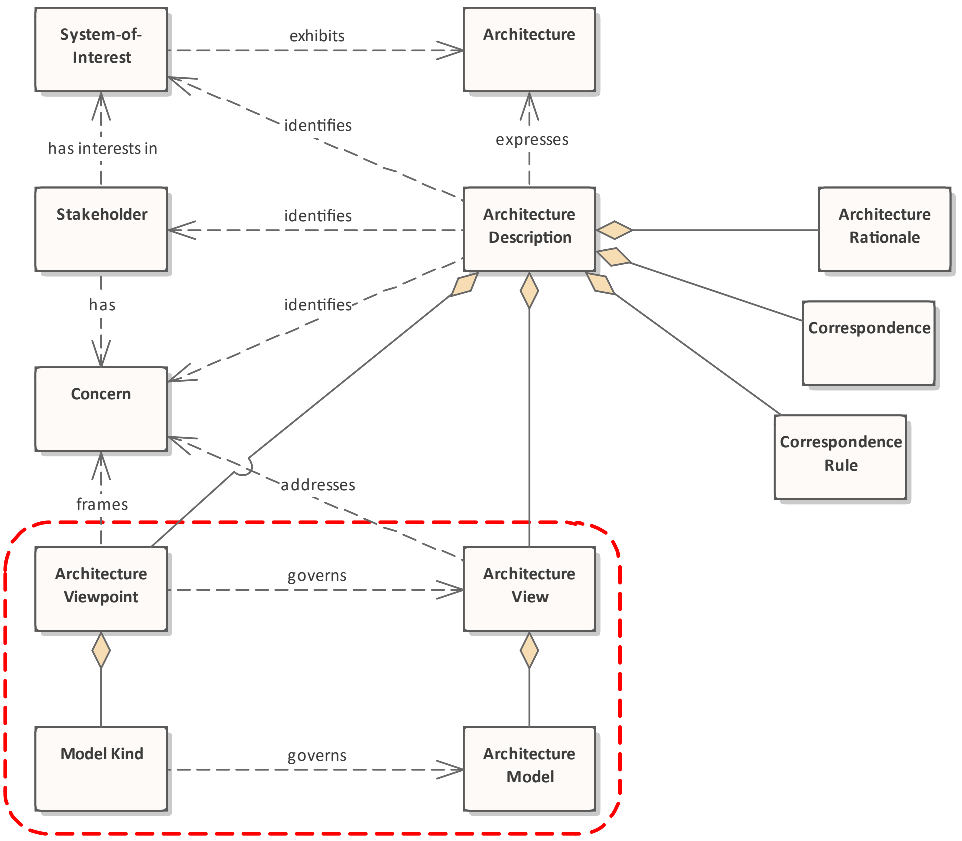


Figure 2.5: ISO 42010 conceptional model

This ontology of the ISO 42010 ensures the interaction between the individual stakeholders, the architecture description, and the actual system to be considered, the SoI. Since stakeholders have different interests in the system, their derived concerns are considered within the architecture description. As a result, this standard regards the aforementioned separation-of-concerns principle as the system's views represent each concern. Each view is expressed in a suitable way allowing the stakeholders to obtain the required information. For example, this could be a business process model for a project manager, a circuit diagram for an electrical engineer, or a logical architecture for a function developer. To achieve the desired functionality, structure, or expected behavior of the system, languages, notations, model types, modeling methods, or analysis techniques are provided in the viewpoint. In summary, ISO 42010 specifies all information that an architecture framework should contain, which is defined as follows:

- Information identifying the individual framework
- The identification of one or more stakeholders
- The identification of one or more stakeholder concerns
- At least one model kind to describe the viewpoint
- Correspondence rules between the individual viewpoints

Stakeholders and Concerns

About the SoI and its environment, stakeholders have specific interests in the system to be developed. These so-called concerns could be derived from one or more stakeholders. These can accrue throughout the system life-cycle and arise from the system's needs, requirements, design choices, and implementation. There are many different ways to record such a concern. Objectives, dependencies, quality attributes, risks, and other types are given as examples for the classification. Among other things, concrete concerns can be functionalities, properties, and feasibility.

Views and Viewpoints

A view of the system might address one or more concerns. The architecture of the SoI is explicitly described concerning the respective viewpoint. Thereby, the view is determined by the viewpoint. This defines the conventions for creating, interpreting, and analyzing the unique view. As any of those views might be individually specified, system architects shall have an “open mind for a different view” (Metallica, 1992). Moreover, a viewpoint connects the concerns and their actual implementation. Such conventions include architectural languages, model types, and design rules.

Architecture Models

The combination of several architectural models results in a view. Each model contains practices and methods for describing the concern. These conventions are specified by the unique model types and thus determine the respective model.

Correspondence and Correspondence Rules

The so-called correspondence defines the relationship between the individual architectural elements. This especially addresses the interest of the individual elements and their connection. The associated rules are used to enforce the respective relationships to ensure integrity.

Architecture Description Languages

Another core element of ISO 42010 is ADLs. These languages are required to narrow the specific concerns and thus advance architecture development through interdisciplinary systems engineering. A well-known example of such an ADL is SysML (Friedenthal et al., 2014).

Once a system based on ISO 42010 has been described, it can be used for several activities. In (Tschirner, 2015), reference is made to MBSE, which is seen as a critical approach to developing the architecture in such systems. Furthermore, it is confirmed that the standard significantly benefits the stakeholders over the entire life cycle. Some examples that support these statements are:

- An architecture description forms the basis for system development and all development activities.
- Various implementation options can be evaluated based on the architecture.
- The architecture of industrial systems can be used for product analysis or simulation tools.

- System architectures can contribute to communication between the individual project participants and the customer.
- A standardized architecture validates the development documentation and provides the basis for successful project management.

2.3.3 Functional Architecture for Systems

Due to a lack of common approaches for developing functional architectures, especially in the context of MBSE, the desire for such a method has become increasingly obvious. Thus, the Functional Architecture for systems (FAS) method has been introduced in (Weilkiens et al., 2015). It provides a methodology for developing a technology-independent, function-oriented description of the system as a block-oriented structure. The main reason this method needs to be applied in modern systems engineering is the upcoming complexity. More precisely, in complex systems, usually, a function is deployed on many physical components, and a physical component realizes more than one function. Therefore, the functional architecture can be interpreted as an interface between the requirements and the physical architecture. By doing so, the FAS provides three main modeling elements:

- Functional Element: This is described as an abstract system element that defines a relation between at least one input and output using a function.
- Functional Group: In terms of FAS, this is a set of strongly related use case activities.
- Functional Interface: As the name assumes, this element defines a set of inputs and outputs of a Functional Element.

Although the FAS method is independent of any modeling language, it is recommended to use SysML for its implementation due to integration opportunities and its acceptance by the community. By doing so, this methodology follows a simple process. At first, use cases and activity diagrams describe all functional requirements' behavior. Subsequently, all strongly related activities are summarized into Functional Groups, which could contain actions or not refined functional requirements. Having elaborated all grouped activities, the next step is to trace each Functional Group into one Functional Element, which builds the base for developing the functional architecture. However, by utilizing SysML, these elements' interconnection and interfaces can be displayed with a block definition or an internal block diagram.

2.4 Systems Application & Validation

2.4.1 System Life-Cycle

The development of a system needs to be structured according to the phases of its life cycle to prevent confusion, misunderstanding, or even conflict, according to Lake (1997). He defines four steps, each dealing with another aspect of the system: Acquisition, Project, Development, and Product. Furthermore, the life cycle of systems development is seen as the most critical task for engineering a system. Its utilization combines results and aspects of several life cycles for optimized decision-making and defining technical efforts. Hence, several sub-processes have been introduced to provide decision-makers with information, starting from designing the system and then developing and evaluating it. About this early approach, today's widely used standard ISO 15288 (International Organization for Standardization, 2015) is the primary outcome resulting from the findings of Lake's development process, which uses the initially introduced spiral model. Through slightly adopted phases and the integration of a V-model, this standard introduces an optimized method for developing a system with the ultimate goal of achieving customer satisfaction. One significant advantage of this framework is the possibility of adapting it to individual application areas by using only those parts needed for developing a specific system. Furthermore, by introducing the V-model, there is a process for evaluating every aspect of the system created as a result of executing the so-called technical processes on different abstraction levels of the system. The structure of the various stages and their respective processes are visualized in Figure 2.6.

Multiple research projects have dealt with using ISO 15288 in actual industrial projects. In more detail, Xue et al. (2017) investigated the optimization of product development in the industrial domain by aligning ISO 15288, which primarily targets systems engineering, with the PMBoK guideline for project management. The outcome thereby shows that the respective processes and activities are covered by each other. However, the systems focused on by the ISO 15288 are mainly products or services, whereas PMBoK primarily targets whole projects. Additionally, when being applied, different development strategies, chronological processing versus concurrent steps, are utilized. All in all, it is claimed that each process model could support the other. The project management processes of ISO 15288 could be replaced by the entire PMBoK guild, while the technical processes of ISO 15288 to complete project management. In addition, another work implements the ISO 15288 for model-based designing of a submarine, which indicates the applicability and suitability of this standard for MBSE (Pearce and Hause, 2012).

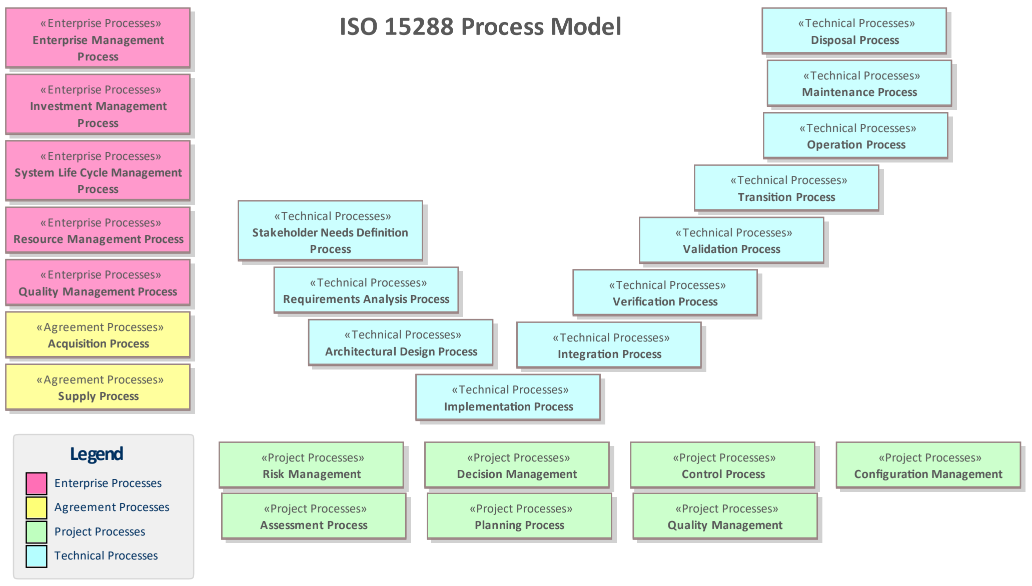

ISO 15288 Process Model
«Enterprise Processes» Enterprise Management Process
«Enterprise Processes» Investment Management Process
«Enterprise Processes» System Life Cycle Management Process
«Enterprise Processes» Resource Management Process
«Enterprise Processes» Quality Management Process
«Agreement Processes» Acquisition Process
«Agreement Processes» Supply Process
«Technical Processes» Stakeholder Needs Definition Process
«Technical Processes» Requirements Analysis Process
«Technical Processes» Architectural Design Process
«Technical Processes» Implementation Process
«Technical Processes» Integration Process
«Technical Processes» Verification Process
«Technical Processes» Validation Process
«Technical Processes» Transition Process
«Technical Processes» Operation Process
«Technical Processes» Maintenance Process
«Technical Processes» Disposal Process
«Project Processes» Risk Management
«Project Processes» Decision Management
«Project Processes» Control Process
«Project Processes» Configuration Management
«Project Processes» Assessment Process
«Project Processes» Planning Process
«Project Processes» Quality Management
Legend
Enterprise Processes
Agreement Processes
Project Processes
Technical Processes

Figure 2.6: ISO 15288 processes

On the contrary, the Open Group originally proposed an enterprise architecture including a utilizable methodology as well as providing a corresponding framework. Better known by the term The Open Group Architecture Framework (TOGAF), the standard has recently been published with version 9.2, as depicted in Figure 2.7. More precisely described in the Foundation Study Guide (Harrison, 2018), the architecture consists of several modular parts.

Requirements management is considered the most fundamental part of enterprise systems engineering, as it is located at the center of the development cycle. Nevertheless, many factors deal with architecture development from Business to Technology and finish with planning the actual migration of the system. Furthermore, some features govern the architecture and design new architectural adjustments.

In conclusion, TOGAF represents an architecture framework inheriting a set of tools that allow the development of various system architectures. According to Harrison (2018), the architectural framework of TOGAF consists of:

- A method for defining an information system in terms of a set of building blocks.
- Indicate the dependencies and connections between the building blocks.
- A set of tools.
- A common vocabulary.
- A list of recommended standards.
- A list of compliant products that can be used to implement the building blocks.

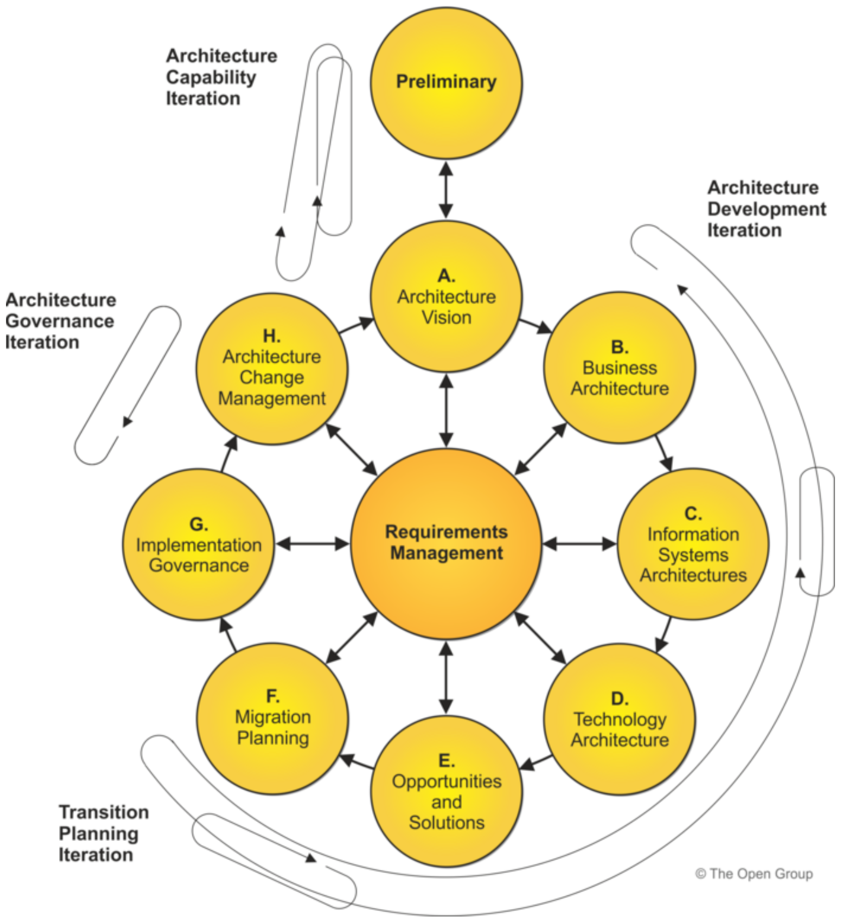

Architecture Capability Iteration
Preliminary
Architecture Development Iteration
Architecture Governance Iteration
A. Architecture Vision
H. Architecture Change Management
B. Business Architecture
G. Implementation Governance
Requirements Management
C. Information Systems Architectures
F. Migration Planning
D. Technology Architecture
E. Opportunities and Solutions
Transition Planning Iteration
© The Open Group

Figure 2.7: TOGAF iteration cycles (The Open Group, 2022)

Therefore, the main pillars of TOGAF are enterprise architecture domains, an architecture development method, and an enterprise continuum. Specifically, the domains address business architecture, data architecture, applications architecture, and technical architecture. Business architecture defines the organization's business strategy, governance, organization, and key business processes. However, the data architecture describes the structure of an organization's logical and physical data assets and the associated data management resources. Additionally, the application's architecture provides a blueprint for the individual systems to be deployed, the interactions between the application systems, and their relationships to the organization's core business processes with the frameworks for services to be exposed as business functions for integration. At last, the technical architecture describes the hardware, software, and network infrastructure needed to support the deployment of core, mission-critical applications. As far as the development method is concerned, in TOGAF, this is represented by its typical iterative cycle. At the same time, the enterprise continuum deals with utilizing architectural patterns, architecture descriptions, and other artifacts to classify architectural solutions. (The Open Group, 2022)

2.4.2 Engineering Toolchain

New trends coming from industry 4.0 are not solely addressing single-running software or autonomous devices. Still, rather than architectural considerations and the complete technological infrastructure around the entire engineering process, the interoperability of tools needs to be ensured within the holistic architecture framework. The toolchain is a collection of tools and their corresponding interfaces organized in chain-based or parallel structures. Each tool of the toolchain might be substituted or replaced with any other tool that provides the same input and output interfaces (Kulcsár, Tatara, et al., 2020). This means the tools within the toolchain build on each other and use each other's output to process this information and generate new output. Thereby, each tool of toolchain could serve a particular purpose, like architecture modeling in terms of a modeling tool or simulation as the task of a simulation tool. Alongside the interface between the tools, each tool might have different outcomes. For example, when considering a machine learning algorithm, the input is provided by the previous tool while the output is passed to the subsequent tools. However, additional information might be only available to users or system optimizers, which deliver input to the algorithm or profit from its output, which is not available to other tools within the toolchain.

Thus, Kulcsár, Tatara, et al. (2020) propose an approach for Industry 4.0 toolchain modeling based on the Arrowhead Framework. By doing so, it is clearly defined what is considered to be a tool to qualify for the toolchain. While the Arrowhead Framework offers multi-purpose tools to test systems, deploy them or detect components, an abstract toolchain has been elaborated. This toolchain consists of the following modeling tasks:

1. Requirements

2. Functional Design
3. Procurement & Engineering
4. Deployment & Commissioning
5. A Operation & Management
6. Maintenance
7. Evolution
8. Training & Education

Additionally, this work was extended to also be applicable within a SoS scenario (Kulcsár, Koltai, et al., 2020). A more specific workflow has been introduced to close the gap between SoS design and SoS operation. To ensure its utilization, the SoS model must be enriched with the necessary configuration information. A suitable file format must be chosen and understandable by modeling and management framework. The workflow is defined as follows: at first, the high-level functional design of a SoS application needs to be created using a particular profile. Next, this information has to be exported and saved in standardized file formats. Thereupon, the file is imported into the management framework via an available interface, enabling the instantiation and subsequent management operation and configuration of the SoS. Finally, the SoS should be exported and imported within the same file format and interfaces to allow changes within the model-based view.

The authors also claim that the toolchain might be applied while designing the system, its run-time, or both. A certain level of automation is required to enable information processing or transfer throughout the engineering processionally; single parts of the toolchain or tools might be used iteratively.

A major contribution within this area has also been proposed by Biffl et al. (2017). This work distinguishes three main phases: the engineering phase, the operation- & maintenance phase, and the end-of-life phase. Various artifacts are introduced that are exchanged over nine separate layers within the engineering phase. Those include artifacts like different layouts, behavioral models, or CAD models. The artifacts used in the operation- & maintenance phase are more informational and include different kinds of plans like an investment, a capacity, or resource management on the upper layers as well as control artifacts like quality, process, and sensor/actor control at the lower levels. Finally, the end-of-life phase introduces mainly information artifacts such as general, component-relevant, material-relevant, or production system-relevant information. However, as standardized information exchange is important for ensuring consistency, AutomationML could be the leading technology for storing single artifacts and their information and transmitting them to other layers or engineering phases. (Biffl et al., 2017)

Due to the increasing integration of different disciplines, the multi-disciplinary design could deal with obtaining an overview of the entire engineering toolchain (H. Li et al., 2019). An example of a fully automated toolchain for MBSE is proposed with automated co-simulation (Lu et al., 2019). The authors present a scenario-based ontology that automatically orchestrates services and generates artifacts usable within a co-simulation. Such artifacts might be entities of a model or a technical resource that is linked to a service. However, the authors claim that ontology serves as a potential specification for realizing simulation automation in general platforms that allow the application of co-simulations.

2.4.3 Co-Simulation

Generally speaking, simulations' main goal is to evaluate system characteristics like controllability, reliability, or functionality. This prevents the need to execute resource-intensive and dangerous laboratory or field experiments. Unlike other simulations, which use either one solver for at least one model or at minimum one solver per model, a Co-Simulation uses multiple solvers for multiple models (Law et al., 2000; Palensky et al., 2017).

Thus, this simulation type is optimized for application on a SoS, such as a IIoT based system. The main advantage of a Co-Simulation is the independent operation of each simulator and the possibility to interconnect them dynamically (C.-W. Yang and Vyatkin, 2018; C.-h. Yang et al., 2013). This interplay, however, can be enabled by using two different kinds of linking. Either the simulators are individually coupled via interfaces with each other or generically with the help of a specific middleware. In the second case, a central unit processes the Co-Simulation scenario during run-time. It deals with exchanging the variables and time synchronization, which is an essential instrument for securing the mentioned aspects by using so-called steps to coordinate each simulator and the whole simulation scenario (Palensky et al., 2017; Schloegl et al., 2015).

To work correctly, a Co-Simulation comprises at least two simulators and a master algorithm, which orchestrates the simulators and manages data exchange or time synchronization. A simulator is a software package or a tool that consists of a model and a solver. However, in the case of a power system simulation, a model contains physical elements and their interconnections. The simulator's job is to transform these system descriptions into equations that the solver can process. Thereby, the output variables of one simulator become the input of one or more other simulators and vice versa (Palensky et al., 2017). This leads to a dynamic coupling of the different simulators that compose the Co-Simulation. To synchronize the outputs of the other time resolutions, fixed exchange times are defined as steps, depending on each simulation scenario. To integrate continuous output values, they are treated as fixed stepped output values with a low step range. Combining different fixed step sizes or variable stepped output values, the tool that manages the synchronization must support the possibility of skipping simulators at specific steps. This differs from integrating event-driven simulators, where an event can only be handled in the next step. In case of inexact results, a reduction of the step size or the implementation of a roll-back function may improve the outcome of the Co-Simulation.

One of those frameworks dealing with middleware has been proposed with Mosaik (Schütte et al., 2011). It was initially developed for the Smart Grid area and has been established and used in several projects (Büscher et al., 2014; Kosek et al., 2014; Steinbrink et al., 2019). The open-source tool is written in Python and integrates a specific power grid simulator like PyPower. To address all aspects a Co-Simulation has to consider, Mosaik comprises four main components. At its core, the Sim Manager is responsible for processing the simulators and their interconnection, while the Scheduler tracks the dependencies between the simulators and performs simulation steps. Additionally, for developing a simulation scenario, the Scenario-API is connected to the core and can be addressed with python code, defined as the Scenario Script. Finally, the framework provides a Sim-API for enabling communication between the simulators. Mosaik designated Component Interface is implemented to manage the communication over plain network sockets via JSON-encoded messages. In contrast to the particular for the IIoT designed project Avanti[1], which is aimed at virtually commissioning and simulating industrial equipment, Mosaik is in constant development. According to these considerations, Mosaik can be considered a hybrid automaton (Henzinger, 2000).

[1]http://www.avanti-project.de/

Chapter 3

Current Concepts regarding Architectures for Production Systems

In contrast to systems engineering, this chapter mainly deals with analyzing frameworks or methodologies for describing system architectures. As industrial systems are classified as complex, such architectural concepts are increasingly important. A comparison between various approaches and utilized frameworks is thus outlined in the following sections.

3.1 Reference Architecture Model Industrie 4.0 (RAMI 4.0)

The three-dimensional model, visualized in Figure 3.1, has been developed by Platform Industry 4.0 to generate a common understanding. This includes standards, use cases, norms, and other relevant aspects of the industrial sector. The scope of this model extends over the entire value-added process and tries to collect and keep consistent technical, administrative, and commercial data. With the other aspects of networking within the company's means of production and the active cooperation of several factories, the discussion of connections and details is made possible using the reference architecture. On the one hand, RAMI 4.0 enables a detailed view of manageable parts of the system, and on the other hand, tasks and processes are displayed across the entire process. This allows a systematic and goal-oriented discussion about classification and standardization. (Bitkom et al., 2015)

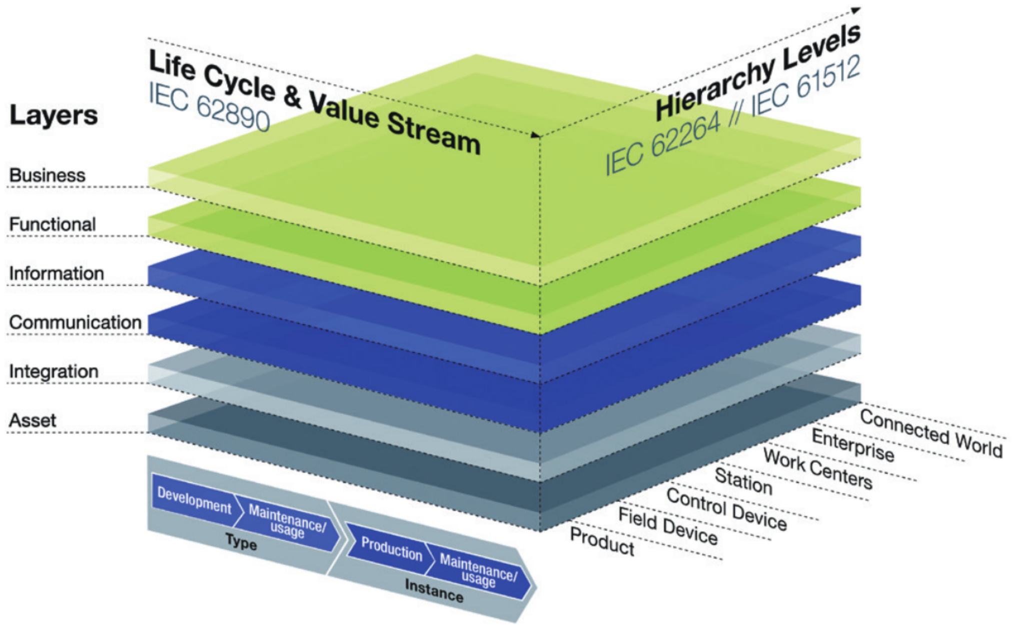


Figure 3.1: Reference Architecture Model Industrie 4.0 (Bitkom et al., 2015)

The figure illustrates that RAMI 4.0 involves three dimensions. The core of the model is the vertically arranged layers. Product development processes and production scenarios are reflected there, structured on six levels. An important characteristic is that there is high cohesion within the layers and loose coupling between the layers. This means that individual logical units only perform their intended tasks and can be dynamically replaced by other teams. The deductive arrangement of these levels is as follows:

Business Layer

The Business Layer defines all the frameworks and rules that the SoI needs to follow. Business processes are analyzed and modeled, and their interactions are shown. Another task at this level is to divide the functions into functional units to prepare them for further processing. No specific devices and systems are specified; this is done in the lower layers. (Bitkom et al., 2015)

Function Layer

The business processes described above must be realized with actual components. The basis for this is laid down in the Function Layer by laying the runtime environment for all services and applications. All functions that exist in the system are prepared and presented in detail here. In addition, decisions about specific scenarios are generated. This is usually done via the Function Layer if the system is accessed from outside. (Bitkom et al., 2015)

Information Layer

The rules generated within the Function Layer are classified and presented in the Information Layer. Events that arise from lower layers are thus prepared for their functional processing. Also, this layer deals with all types of data in the system. These are collected, summarized, and processed to generate new data from them or pass them on to the outside world. Above all, it is essential to ensure the integrity of the data. (Bitkom et al., 2015)

Communication Layer

This layer primarily serves as an interface between the adjacent layers. Constant communication is generated between the individual components. Providing these services via a fixed infrastructure allows data to be exchanged via these paths, and devices based on information and communication technologie (ICT) can be controlled. (Bitkom et al., 2015)

Integration Layer

This layer prepares all physical devices and components for the virtual world. This is done by generating events and processing them in the higher layers. Maintaining a consistent connection with the object layer is essential so that changes, in reality, are fully and promptly recorded. (Bitkom et al., 2015)

Asset Layer

In the end, Industry 4.0-based systems are realized with physical components. Therefore, it is essential to capture every element within the Asset Layer. People and ideas must also be taken into account here. If devices are not capable of active communication alone, they can be connected via a passive connection such as a bar- or QR code. (Bitkom et al., 2015)

In addition to the vertical dimension, two further dimensions are integrated into RAMI 4.0. On the one hand, the life cycle of the modeled systems is taken into account; on the other hand, the functional classification within the value chain is addressed. This creates a horizontal rectangle in which each physical unit occupies a specific place. The second axis of RAMI 4.0 thus determines the current state during the entire life-cycle of a device. The basis for this is the draft for IEC 62890 (International Electrotechnical Commission, 2016b). This states that a distinction between type and instance is of central importance. As soon as there is no concrete unit of a product, from the idea through the development and creation phase to the tests and prototype production, this is considered a type. Only after a manufactured product can it be identified by a serial number, for example, can it be classified as an instance. This change can occur several times to respond to improvements or changes during the development phase. (Bitkom et al., 2015)

The third axis is used to classify the components within the factory. Derived from the two standards IEC 62264 (International Electrotechnical Commission, 2016a) and IEC 61512 (International Electrotechnical Commission, 2001), it specifies a classification scheme. This ranges from the connected world via the enterprise to the respective work units. In addition, individual systems, as well as devices and products, are covered. This allows CPSs to be classified within the model according to functionality and status. (Bitkom et al., 2015)

The comparison of RAMI 4.0 to other reference architectures has shown that this kind of reference architecture model is very generic. It offers a good overview of all key concepts within a Smart Factory. However, doing so leads to some limitations regarding understanding the exact positioning of different technologies and functions and their connectivity. In more detail, planners of smart factories might not know where to place the respective key concepts within the model's layout, apart from even interconnecting them. This hinders the interplay of various digital twins or digital agents, which is considered the main challenge for planners of smart factories. (Resman et al., 2019)

An additional work proposed the alignment of those core concepts used in the context of smart factories to the layers and automation pyramid axis of RAMI 4.0 (Wang et al., 2017). This alignment shows that the life-cycle axis could be treated regardless of when structuring an entire production system. Figure 3.2 thus demonstrates the results of this topological approach, which aligns each concept to the respective two-dimensional scale.

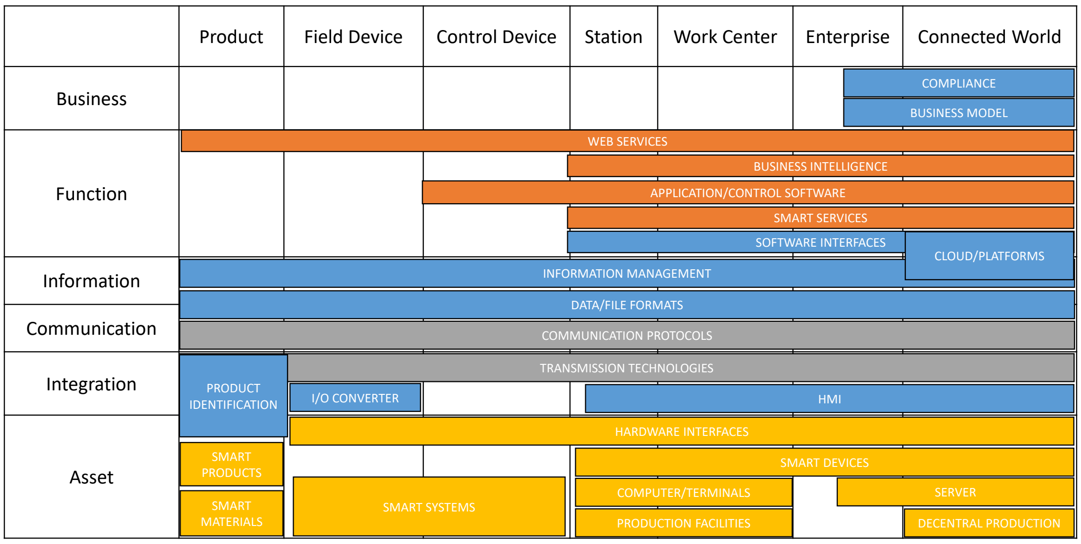


Figure 3.2: Mapping smart factory concepts RAMI 4.0 based on (Wang et al., 2017)

The image shows various concepts and their assignment. For example, business models or compliance could be found at the Business Layer of RAMI 4.0 and the connected world pane within the automation pyramid axis. Subsequently, different types of services, like web or intelligent services, and the corresponding control software are assigned at the Function Layer. About the automation pyramid, those concepts interconnect the connected world within the factory compartments, like workstations or work centers. Additionally, business intelligence is also considered within this layer. Software interfaces represent the interfaces between the Function Layer and the Information Layer at more minor granularity levels and platforms or the cloud at higher granularity levels. However, the central concept of the Information Layer is information management, which spans all hierarchy levels of the factory. The same counts for communication protocols and transmission technologies, which are placed at the RAMI 4.0 Communication Layer. File formats or data thereby represent the interface between

Information and Communication Layer.

Subsequently, the Integration Layer mainly deals with providing human machine interfaces (HMIs) at higher granularity levels, IO-converters at the field device level, and product identification at the product level. Finally, the Asset Layer consists of various intelligent factory components, like smart products, smart devices, intelligent systems, computers, servers, or other facilities, depending on the hierarchy level within the automation pyramid. However, this topological classification of RAMI 4.0 builds the base for future assignments of key concepts into the three-dimensional model and thus for specifying metamodels, DSLs or viewpoints and model kinds.

3.1.1 Asset Administration Shell

In RAMI 4.0, each virtual and physical component is encapsulated in an asset administration shell, which is used to administer one or more elements. For this purpose, several requirements are placed on the asset administration shell, such as clear identification, management of data, and provision of an Industry 4.0-compliant communication interface (Ye and Hong, 2019). Due to these properties, it serves as a central element within a system. Each component must have at least one administration shell to ensure seamless information transfer within the system. To ensure data integrity, the asset administration shell exhibits a unique structure. The head of the asset administration shell serves as an index, which uses the manifest to refer to the individual characteristics where data and functions are stored. Combined, several features from individual units, so-called part models, are managed by a component manager. This structure allows the asset administration shell to include different contents, thus storing central information and passing it on to the outside. Figure 3.3 shows an overview of several topics that can be addressed in this way.

Essential requirements are descriptions to ensure correct working methods and identification and communication. Security and configuration options are also included in its part models. Several standards have been defined for the uniform description of these and other requirements, which are exemplary due to the current state of research (Plattform Industrie 4.0, 2016).

RAMI 4.0 is considered a technology driver in the modeling of IoT systems in the German-speaking area. Through the cooperation of critical German associations and the federal government, it benefits from nationwide support. As a result, it is constantly being further developed, and it is possible to integrate new events or changes in the environment quickly. This dynamic flexibility is essential in a continually changing domain.

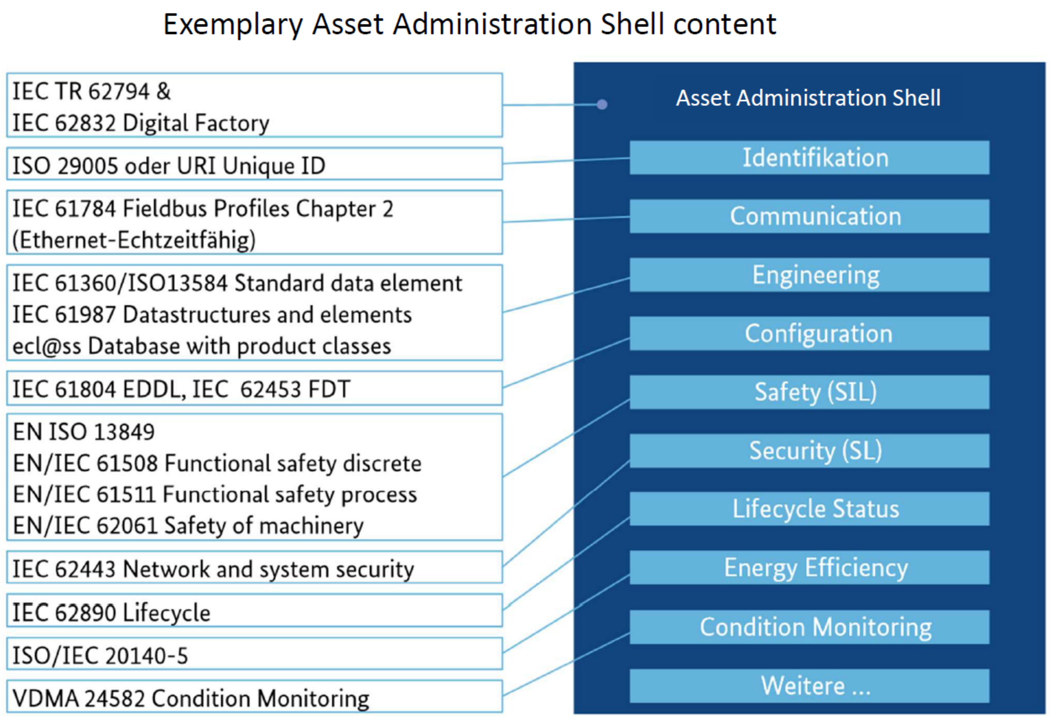

Exemplary Asset Administration Shell content
IEC TR 62794 & IEC 62832 Digital Factory
ISO 29005 oder URI Unique ID
IEC 61784 Fieldbus Profiles Chapter 2 (Ethernet-Echtzeitfähig)
IEC 61360/ISO13584 Standard data element
IEC 61987 Datastructures and elements
ecl@ss Database with product classes
IEC 61804 EDDL, IEC 62453 FDT
EN ISO 13849
EN/IEC 61508 Functional safety discrete
EN/IEC 61511 Functional safety process
EN/IEC 62061 Safety of machinery
IEC 62443 Network and system security
IEC 62890 Lifecycle
ISO/IEC 20140-5
VDMA 24582 Condition Monitoring
Asset Administration Shell
Identifikation
Communication
Engineering
Configuration
Safety (SIL)
Security (SL)
Lifecycle Status
Energy Efficiency
Condition Monitoring
Weitere ...

Figure 3.3: Asset administration shell content based on (Plattform Industrie 4.0, 2016)

3.1.2 Service-oriented Architectures

Due to the integration of IoT-devices into systems, the importance of service-oriented architectures (SoAs) is constantly increasing. Thereby, services are valuable concepts, as they can be considered from different perspectives. Figure 3.4 illustrates the four main views a service could implement. The business perspective outlines what customers are willing to pay for, while the technology perspective indicates, what is to be implemented, but not how it is implemented. Furthermore, the value proposition for consumers is stated in the same-mentioned perspective, while the provider perspective further explains the provision of the service by prevalent technology solutions. (Perrey and Lycett, 2003)

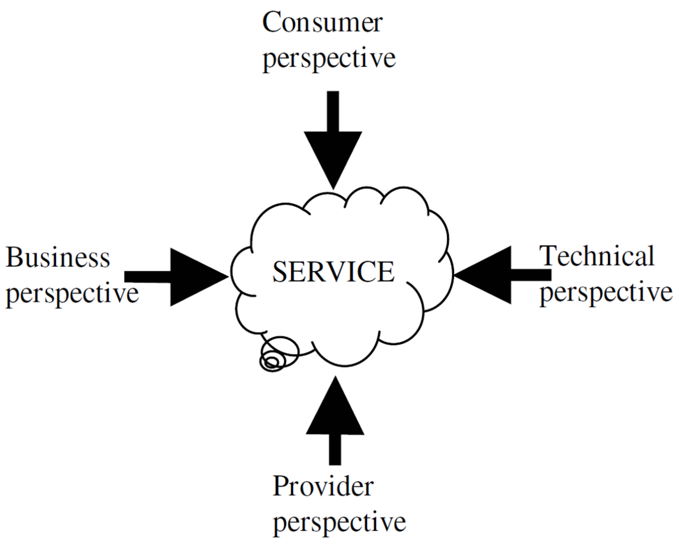


Figure 3.4: Perspectives on services (Perrey and Lycett, 2003)

To mention an example of successfully implemented SoAs, the authors of (Mendoza-Pitti et al., 2021) analyze such kind of architecture to describe the energy efficiency in Smart Buildings. As building management systems need to be able to monitor, control, research, and manage the components used within a Smart Building, integrating many proprietary devices is becoming increasingly difficult. A SoA thereby helps to identify the interconnection and the data exchange between those devices by specifying the functionalities of each device as services.

More specifically targeting the Industry 4.0 domain, a detailed analysis of SoAs for manufacturing systems is published in (Reis and Gonçalves, 2018). In this work, the Internet of Services is tried to be delimited, and the effects on the manufacturing environment are investigated. The authors conclude that the Internet of Services is one pillar of Industry 4.0, as each manufacturing element provides its functionality as a service. Thus, a SoA could characterize this manufacturing system better.

A more detailed approach has been proposed in (Schnicke et al., 2020). The minimal needed data to enable service orientation in such a manufacturing system can be derived by defining Digital Twins of actual physical components. The services themselves are analyzed towards their quality concerning time, money, and product quality as well as the capabilities the service can fulfill. The needed data is gathered from two use cases describing manufacturing plants by modeling the customers' orders and the products to be manufactured. While making the first steps of defining a SoA in this direction, the proposed approach appears promising when modeling the services of Digital Twins.

Other approaches dealing with the service-oriented development of industrial systems also make clear that the action of SoAs in this area is needed. For example, another approach also uses RAMI 4.0 to retrofit original manufacturing systems towards Industry 4.0 (Contreras et al., 2017), while the authors of (Liu et al., 2020) define a model-driven development process based on the Reference Model for Service Architecture (RM-SA). In addition, some publications also introduce new SoAs for Industry 4.0-based systems (Al-Jaroodi et al., 2018; El Kho and Aknin, 2019). While most of the mentioned approaches propose initial prototypes and frameworks in this area, they mostly solely address one single aspect of the manufacturing system or target a specific domain. A holistic and standardized approach must be available to enable the mutual engineering of current and future industrial systems, including multiple stakeholders and considering many fields.

3.2 State-of-the-Art in Industrial Systems Engineering

As already stated, other projects used MBSE to develop aspects of flexible production systems. First applications of MBSE to handle the increasing complexity within industrial embedded systems have been proposed in (Sindico et al., 2012). By investigating the methodology utilizing a radar application, the authors claim that MBSE is a promising technology driver for developing complex heterogeneous industrial systems. To mention another example, Tolio (Tolio, 2008) addressed this topic more than ten years ago by providing an overview of methodologies and tools for designing flexible production systems. While MBSE has barely been a topic at the early stages, more recent publications recognized the possibilities of this methodology to deal with the upcoming system complexity and improve manufacturing system flexibility. More precisely, in (Tliba et al., 2020), the development of a flexible production system in the perfume production area has been supported by using MBSE. The authors claim that different stages of the development process, like scope definition, needs analysis, SysML modeling, and requirements derivation, could design a particular decision-making system. By evaluating the result with a perfume production case study, it has been shown that MBSE helps to find best-fitting solutions for desired flexibility dimensions and relevant associated criteria.

Another work also states that MBSE is the leading technology driver for developing SoS in the area of industrial systems (Schluse et al., 2018). By creating so-called experimental Digital Twins of production systems, single components of the system or the system itself might be optimized by utilizing simulations. Therefore, the authors state that such experimental digital twins should be considered throughout the whole system life-cycle, from its engineering with MBSE up to its actual usage. A collection of related trends is thereby proposed in (Akundi et al., 2021), where the authors identified often used tools and languages when considering MBSE within the area of production engineering.

However, the first attempts to advance digitalization in production systems were published in (Grangel-González et al., 2016). There, the modeling of a digital twin for a CPS is proposed using the asset administration shell of RAMI 4.0, whose standards and technologies are used. Further work resulted in the publication of a knowledge graph, which analyzes data in semantic interoperability scenarios and integrates them into these systems (Grangel Gonzalez, 2019). Well-known industry standards such as AutomationML or Open Platform Communications Unified Architecture (OPC UA) are used and entered into these knowledge graphs together with the CPSs. The well-known standardization frameworks such as RAMI 4.0 and Industrial Internet Reference Architecture (IIRA) are also integrated into these graphs. The work results show that knowledge graphs are suitable for using data in Industry 4.0 scenarios to solve semantic interoperability conflicts.

Another project deals with data analysis in industrial factories with its architecture framework (Arantes et al., 2018). SysML is used to enable MBSE, and particular emphasis is placed on traceability and quality in the models produced. This makes it easy to implement changes in the model. The architecture framework includes solutions for extensive data analysis, data flows, and user interfaces. Other approaches, which also use architectures based on RAMI 4.0, are used on the one hand to find equipment to fulfill process operations (Pisching et al., 2018); on the other hand, the integration of security is promoted (Sharpe et al., 2019). A different approach describes continuous and integrated modeling and analysis of relevant information within an Industry 4.0 system (Mandel et al., 2020). The dependencies and relationships in such a production system are examined more closely and then presented with model-based approaches. The result of this project states that information is more valuable the earlier it is available to all participants within the value-added process. The information and its dependencies are modeled with SysML.

As far as RAMI 4.0 is concerned, several approaches have recently been published trying to understand the implementation of the reference architecture for industrial systems based on case studies (Barbie et al., 2020; Jeon et al., 2020; Lins and Oliveira, 2020). As most of these projects indicate the direction of future research, some other works have already had a larger influence on the community (Pisching et al., 2018; Yli-Ojanperä et al., 2019). However, as most of the mentioned propositions focus on implementing RAMI 4.0 practically, it becomes obvious that the standardized reference architecture is missing formulations for actual industrial applications. In contrast, based on IIRA, an approach has been published that enables MBSE and treats IIoT-based systems as SoS. This counteracts the problem that there are few formal approaches in this area. To master the complexity of such an extensive system, the methods and viewpoints of IIRA were expanded with those of Unified Architecture Framework (UAF). Hence, this creates a comprehensive architecture that incorporates the advantages of both approaches. Based on a household appliance case study, it was evaluated whether UAF is suitable for such an application. (Morkevicius et al., 2017)

Apart from MBSE, a more general approach proposes an architecture for Digital Twins of flexible production systems (Talkhestani et al., 2019). The architecture is divided into a physical layer and a cyber layer. In contrast, the cyber layer consists of several models of the physical asset, assuring simulatability, active data acquisition, and synchronization. At the top, artificial intelligence (AI) is also considered within the introduced architecture. An example of using machine learning to optimize CPS with data and learning algorithms to make intelligent decisions is outlined in (Villalonga et al., 2020). Thinking ahead, such an architecture of a Digital Twin could be located in the Integration Layer of RAMI 4.0. Other approaches proposing architectures for such flexible production systems mainly targeted to RAMI 4.0 can be found in (Fan et al., 2021; Rementsov and Lukinov, 2020).

Finally, a summary of the analysis of different modeling languages in the industrial sector is published in (Wortmann et al., 2020). The publication explains that almost half of all documents in this area are published in Germany. A large part of the published papers was presented at the Emerging Technologies and Factory Automation (ETFA) and Industrial Informatics (INDIN) conferences, which offer their technical tracks for this domain. It is also significant that the number of publications in this area has only surpassed 20 per year since 2012 and has increased sevenfold by 2017. Most of the topics addressed are digital representation, system integration, and processes in general.

To give additional examples for using MDA to develop industrial systems, the design of logical controllers based on models is introduced in (Łabiak and Bazydło, 2018). In contrast, wireless sensor networks have been developed utilizing MBSE in (Anwar et al., 2019). The need for efficiently organizing, accessing, and managing MBSE artifacts has also been outlined in (Madni, 2021). By evaluating their results with an aircraft perimeter security system and an adaptive planning and decision-making system for autonomous vehicles, the authors proposed a particular testbed for assessing the MBSE artifacts. Several model-based approaches also deal with analyzing big data (Erraissi and Belangour, 2020) or data visualizations (Golfarelli and Rizzi, 2020). Additionally, automated model transformations have been investigated in detail by making use of MBSE and UML diagrams (Deeba et al., 2018; Melouk et al., 2021; Yurin et al., 2019). Models and transformations save time or other resources while engineering flexible production systems.

The approaches presented in this section imply that MBSE within industrial systems is not a new research topic. However, it is challenging to consolidate a uniform approach due to the different standards and proprietary solutions. Many of the respective projects deal with a single aspect of systems engineering and data analysis. A common and standardized approach can be the key to improving cooperation in this area. Since RAMI 4.0 provides all the tools needed for such collaboration, it is considered a promising approach to achieve this.

3.2.1 PPR Systems

To reduce costs and enhance flexibility in production systems (Schumacher et al., 2016), processes, including multiple domains and models, need to be digitalized. Thus, multi-view models could be the primary technology driver to substantiate the collaboration and exchange of heterogeneous information (Atkinson et al., 2015). By doing so, each domain expert could remain in their area of expertise and follow established habits, where results are expressed in a single viewpoint of the multi-view model. Moreover, to allow those exports to communicate their knowledge on the one hand and transform and provide this information to the whole architecture on the other hand, so-called DSLs are utilized. This results in CPPS Engineering Networks (CENs), where domain-specific and cross-domain data models need to be elaborated dealing as a basis for the digitization of engineering data logistics (Lüder, Pauly, et al., 2019).

However, a sufficiently integrated multi-view system model is a precondition for interconnecting heterogeneous engineering disciplines and their respective domain experts. According to Atkinson et al. (2015), such a model, including multi-domain elements with domain-specific links, is costly and error-prone to create from implicit knowledge. Thus, the need for a modeling framework addressing those domains and providing a needed methodology supporting this task becomes apparent. To enable modeling with AutomationML, multi-view modeling has been proposed by Schleipen and Drath (2009). As AutomationML defines several advanced concepts for various engineering aspects, new engineering practices must be considered. One of these practices is introducing the so-called PPR concept.

However, different views are generated to counteract the complexity of contemporary plant engineering data. This allows consider complex scenarios from varying perspectives and thereby separate concerns or divide complex concepts. The three main viewpoints to be considered within the PPR concept are products, processes, and resources. In more detail, the three views are explained as follows (Schleipen and Drath, 2009):

- Considering the system from a resource view, those are building the center of interest. Resources represent the centric point of view, the main entities involved in production. Resources might execute processes or handle products to find the correlation to the other two views. Resources could be any machine, robot, conveyor, or even software by being modeled as a kind of topology within the plant hierarchy.
- In contrast, from a product-centric point of view, products represent produced goods, which are the central point for consideration. Thereby, products might be processed by any process, from raw material handling to manufacturing intermediate products. When engineering a product, any form to describe them is possible.
- Finally, the processes form the main point of interest within the process-centric point of view. In more detail, the production process with all sub-processes or parameters represents such processes within the process chain. From a technical perspective, a production process modifies products. Therefore, any production process could be considered within the PPR concept, from welding, transporting, or assembling different sub-products.

The single elements of the PPR concept and their interaction are thereby visualized in Figure 3.5. As illustrated in the image, those are interlinked to each other and thus are valid. This interconnection could be described as; a product being manipulated by a resource following a process, defined as a natural concept. Hence, processes are used to produce products, which are produced on resources. To close the link, processes are executed in resources. Thereby, all three views might be represented in separate object hierarchies. In addition, each of the views might have different stakeholders having an interest in the system. (Biffl et al., 2021)

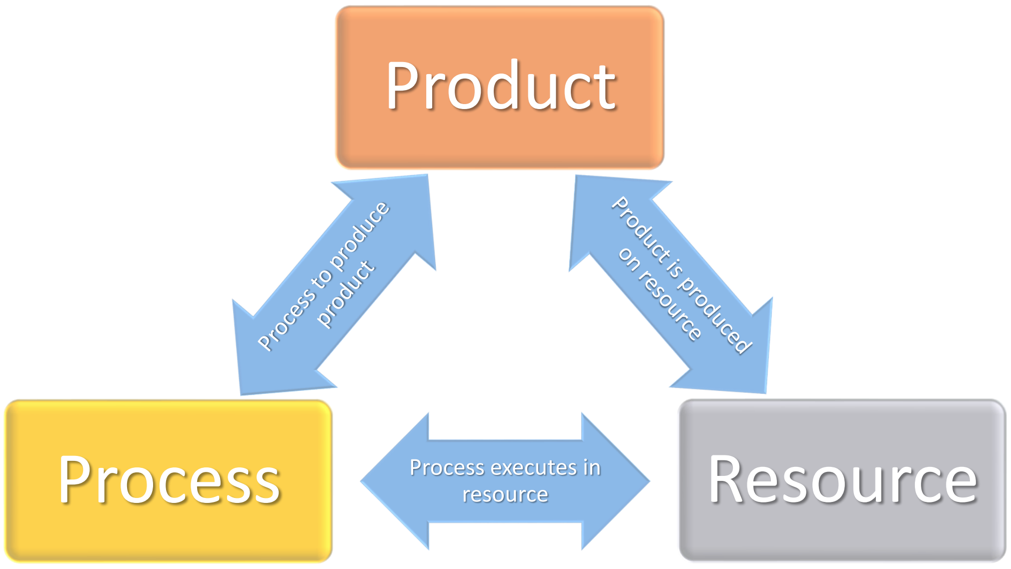

Product
Process to produce product
Product is produced on resource
Process
Process executes in resource
Resource

Figure 3.5: PPR elements based on Schleipen and Drath (2009)

Additional research projects have emerged based on the PPR concept. For example, a follow-up publication (Pfrommer et al., 2013) introduces skills into such systems. Thereby, products and resources keep their original meaning, while processes might be considered abstract skills. Those are independent of products and resources by denoting generic capabilities. By being useful in a production setting, many of such process examples are listed in the DIN 8580 (Förster and Förster, 2018). Within this standard process, hierarchies are considered, where sub-process derive from more general descriptions. To interconnect these processes with resources, the term of the skill is introduced (Herzog et al., 2020). This term defines the ability of a resource to perform a process. By representing this interrelation, additional information might be added. This is needed as different sub-process might be executed on different resources. The feasibility of the resource to execute the process is thereby extracted from the enriched information. Finally, to summarize the PPR concept, tasks are introduced to define the application of a skill on a specific type of product, aiming to achieve a desired outcome. Therefore, tasks represent the relation between a product and a skill, which implies the definition of the executing resource. From an engineering perspective, tasks could be considered functions with pre- and post-conditions and temporal constraints. (Pfrommer et al., 2013)

3.2.2 Basic Engineering - Detailed Engineering

Various process structures have been proposed to develop production systems by executing engineering processes as they are partially standardized and barely interconnected. However, revealing similarities, an overview of existing engineering processes is given in (Lüder et al., 2011). Thereby, two significant standards need to be discussed when considering a solid distinction between the basic engineering of production systems and their detailed engineering. At first, the VDI 5200 Guideline for factory planning has been proposed to formalize this process (Heinen et al., 2010). Using all relevant engineering information as a base, the process is considered a controlled information enrichment process, including eight phases. By inheriting single steps for preparing, finalizing, or implementing the system, a significant distinction is made between the concept design and the detailed planning, representing an analogy for basic and complex engineering. A more compact solution has been introduced with the VDI 3695 Guideline, which addresses engineering efficiency and quality within the plant engineering process. Engineering artifacts are exchanged over five phases, interconnecting plant development with product design (Verein Deutscher Ingenieure, 2009).

However, the VDI 2206 Guideline uses the well-known V-model to develop mechatronic systems. It thus might be one of the most promising process models when dealing with MBSE in the area of Industry 4.0. More specifically, as visible in Figure 3.6, the development process inherits six different phases (Gausemeier and Moehringer, 2002):

- Problem Description: Within this phase, the requirements for the intended product are collected. Thus, the defined object is more precisely specified and described as requirements.
- System Design: This phase describes the development of the overall product structure by defining the system component hierarchy as well as its interfaces. The aim is to create a cross-domain solution concept describing the future production system's main physical and logical operating characteristics. By doing so, overall system functions are hierarchically decomposed into sub-functions and evaluated in the context of the system.

- Detailed Engineering: Domain-specific engineering in different engineering disciplines like mechanical, electrical, or information technology is the main focus of this phase. Thereby, all documents for detailed engineering are created similarly. Critical functions are interpreted in more detail to ensure the system's performance.
- System Integration: The results of each engineering domain are collected and integrated into a complete product within this phase.
- Validation: The fifth phase deals with validating the created product by considering the requirements of the first phase.
- Modeling and Analysis: The overall process model is supported by this phase, which aims to model the intended product, its properties, and its capabilities.

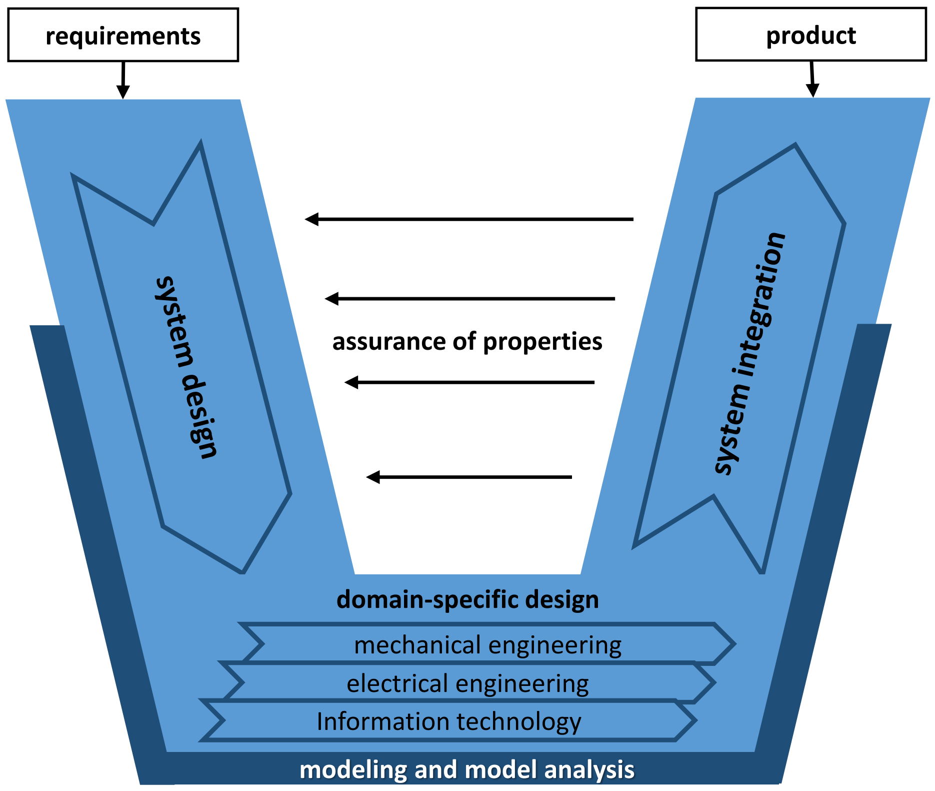


Figure 3.6: VDI 2206 V-model based on (Gausemeier and Moehringer, 2002)

In conclusion, this means that the VDI 2206 Guideline strongly focuses on basic engineering and detailed engineering. As different processes and tasks need to be fulfilled in one of the respective disciplines, particular modeling methodologies need to exist. As previously described, basic engineering mainly specifies system functions and structure. Thus, the main tasks in this phase are factory planning or the layout of elements. Additionally, logical architectures consider different system concepts, while physical elements represent actual implementations. Those physical elements, which also realize the functions, are then passed to the domain-specific engineering phase and refined.

Based on the documents created by basic engineering activities, detailing is subsequently done by detailed engineering activities. This type of engineering includes the coordination between all considered areas and domains of engineering, like mechanical, electrical, functional, or information technology. The results of interdisciplinary engineering are presented in several documents, such as specification sheets, equipment, and schedules. Revisions in design or operations and maintenance are also tasks to be considered within the detailed engineering phase. (Chakrabarti, 2022)

In more detail, mechanical engineering is described as a professional engineering discipline involving applying principles from physics, design, manufacturing, and maintenance of mechanical systems. A solid understanding of key concepts within mechanics, kinematics, thermodynamics, and energy is required to enable this engineering discipline. Main engineering disciplines that utilize mechanical engineering as systems engineering discipline could be specified industrial equipment or machinery and the design and analysis of automobiles or aircraft. In Industry 4.0, mechatronics and robotics are valuable principles to consider during mechanical engineering. (Grote and Antonsson, 2009)

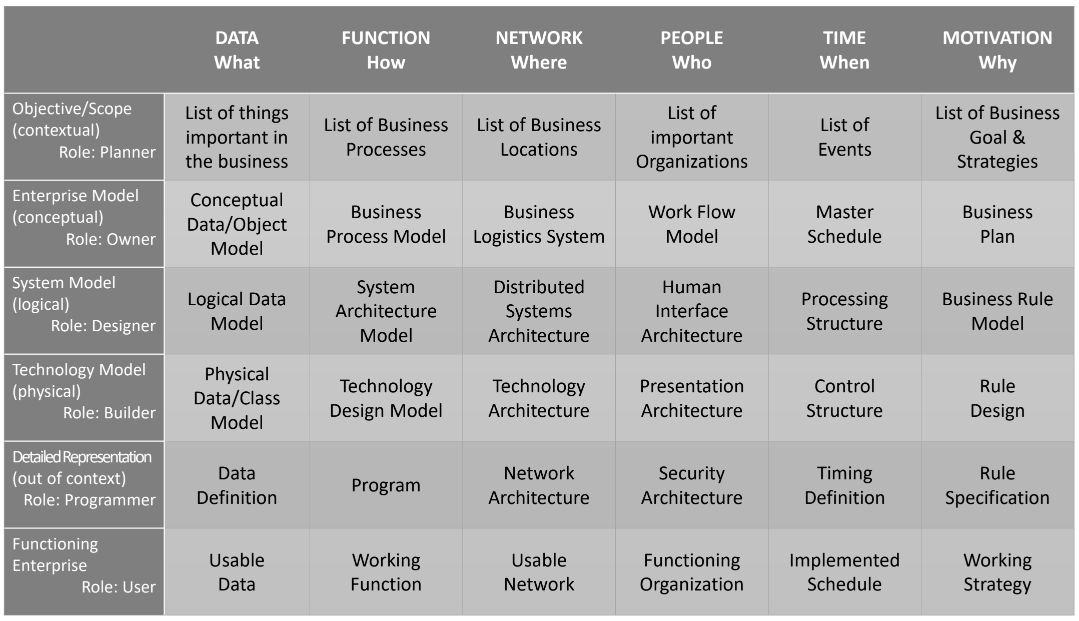
On the other hand, circuit theory is one of the main disciplines to be considered when performing electrical engineering. Hence, a circuit is represented by an interconnection of electrical elements, including passive elements like resistances, capacitances, and inductances. Active elements extend those elements to provide functionality for the circuit. However, each element contains two main variables, voltage, and current. This means when performing systems engineering within the electrical domain, the analysis and design of circuit diagrams are mainly addressed. (Chen, 2004)

As far as information technology is confirmed, new possibilities for implementing those system components into organizational structures need to be elaborated. Orlikowski and Robey (1991) explain that information technology components should be positioned centrally within the process of structurization. This allows for fostering the relationship between organizations and information technology, which is deployed into a company to accomplish a determined task like creating, recreating, or transforming human interaction by providing an objective set of rules.

3.3 Architecture Frameworks

3.3.1 Zachman Framework

It was recognized early on that information systems were growing in size and complexity. An attempt was made to develop architectures for the logical system decomposition to integrate all components into the system and manage their interfaces. For this reason, the so-called Zachman Framework was proposed by John Zachman for IBM and published at the end of the 1980s (Zachman, 1987). In its original form, the architecture was solely described by handwritten drawings. However, the framework has evolved over the years into a complete and comprehensive methodology for MBSE through several adjustments. As visualized in Figure 3.7, the structure of the method is based on a matrix. The Zachman Framework specifies which stakeholder is addressed by which artifact and compares it to the original problem. Thus, this framework is still relevant for modern systems since information is spread across the entire company.



	DATA What	FUNCTION How	NETWORK Where	PEOPLE Who	TIME When	MOTIVATION Why
Objective/Scope (contextual) Role: Planner	List of things important in the business	List of Business Processes	List of Business Locations	List of important Organizations	List of Events	List of Business Goal & Strategies
Enterprise Model (conceptual) Role: Owner	Conceptual Data/Object Model	Business Process Model	Business Logistics System	Work Flow Model	Master Schedule	Business Plan
System Model (logical) Role: Designer	Logical Data Model	System Architecture Model	Distributed Systems Architecture	Human Interface Architecture	Processing Structure	Business Rule Model
Technology Model (physical) Role: Builder	Physical Data/Class Model	Technology Design Model	Technology Architecture	Presentation Architecture	Control Structure	Rule Design
Detailed Representation (out of context) Role: Programmer	Data Definition	Program	Network Architecture	Security Architecture	Timing Definition	Rule Specification
Functioning Enterprise Role: User	Usable Data	Working Function	Usable Network	Functioning Organization	Implemented Schedule	Working Strategy

Figure 3.7: Zachman Framework based on (Zachman, 1987)

The Zachman Framework is defined in different ways (Wikipedia, 2021):

- A framework for organizing and analyzing data
- An enterprise architecture framework
- A classification system or scheme
- A matrix in 6x6 format
- A two-dimensional analytical model
- A two-dimensional scheme for organizing detailed representations of the company

Hay (1997) also describes how the individual rows of the matrix are defined:

Objective/Scope

To support the planner of the system, this row defines the direction of the enterprise and its purpose. Based on this, the requirements can be derived from the context of the system to be created by analyzing the company's environment with an industrial reference. (Hay, 1997)

Enterprise Model

The administration of the company is mainly concerned here. The nature of the company, its structure, and its organization, as well as its functions, are described in detail. (Hay, 1997)

System Model

The company is also the focus within this row, but with more reference to information. This should help the system architect use the functions of the row above, thereby ensuring sustainable data storage. Specifying which information the company wants to collect and organize is important. (Hay, 1997)

Technology Model

This row indicates how the previously described information is technically processed. The designer determines which databases are used, how the program structure is defined, and which user interfaces and languages are used. (Hay, 1997)

Detailed Representation

The system's components are implemented based on the previously selected technologies. The programs, databases, networks, programming, and query languages selected for this are displayed within this row. (Hay, 1997)

Functioning Enterprise

Finally, the whole system is implemented and used by being incorporated into the corporate structure. (Hay, 1997)

On the other hand, each column provides a different viewpoint to classify the company's diverse information. The data column describes the company's available data and how it is distributed to other information carriers. The individual rows are used to decide which components must be present to organize them and what the actual use can look like in detail. Additionally, the processes for processing this data are shown in the function column, while the network column deals with the local distribution within the company. In addition, selected technologies and network components are modeled in this column. The last three columns decide which people are involved in the process, when and why tasks are carried out, and the motivation and the actual reason for the process. (Gerber et al., 2020)

Since the architecture of the Zachman Framework only describes which system components can be found where and how they are related, additional methods must ensure the correct matrix description. The goal should be to avoid redundancies as much as possible and to consider both physical objects and conceptual ideas (White, 2020). The resulting architecture should ideally provide a clear picture of the business organization and information architecture. For this reason, the Zachman Framework gives seven design rules or principles as guidelines for filling the two-dimensional matrix (White, 2020):

1. The individual columns do not have a precise order but should be arranged top-down. It starts with the most significant category. Depending on the project, this might change, but all fields in the matrix should always be filled out.
2. Each column has a simple generic model and possibly its metamodel.
3. The basic model in each column should be unique and avoid overlapping or data replication.
4. Each line describes a unique perspective. Metamodels affecting multiple cells should be avoided to avoid redundancies within the matrix.
5. After the rules mentioned above are executed, each field within the matrix is unique, resulting in a detailed and informative view of the architecture.
6. The names of the rows and columns should not be changed, as this can lead to confusion or inconsistency.
7. The logic within the Zachman Framework is consistent and generic. This makes it possible to classify and analyze every single aspect of the company. The framework user should determine the scope and boundaries for a specific project, as this can significantly impact the matrix, initiative, or project outcome.

3.3.2 Software Platform Embedded Systems (SPES)

As the name assumes, the Software Platform Embedded Systems (SPES) modeling framework is used for the model-based development of different kinds of systems. Thus, the framework represents a set of tools enabling the model-based development of different systems in several areas. By doing so, SPES challenges originate from different application domains like automation, healthcare, and automotive, amongst others. Based on specific requirements and principles defined in (Pohl et al., 2016), a new way of thinking regarding systems engineering is proposed. For example, the concept assumes that the system's development characteristics should be derived from the requirements within the specific application domain, as visible in Figure 3.8.

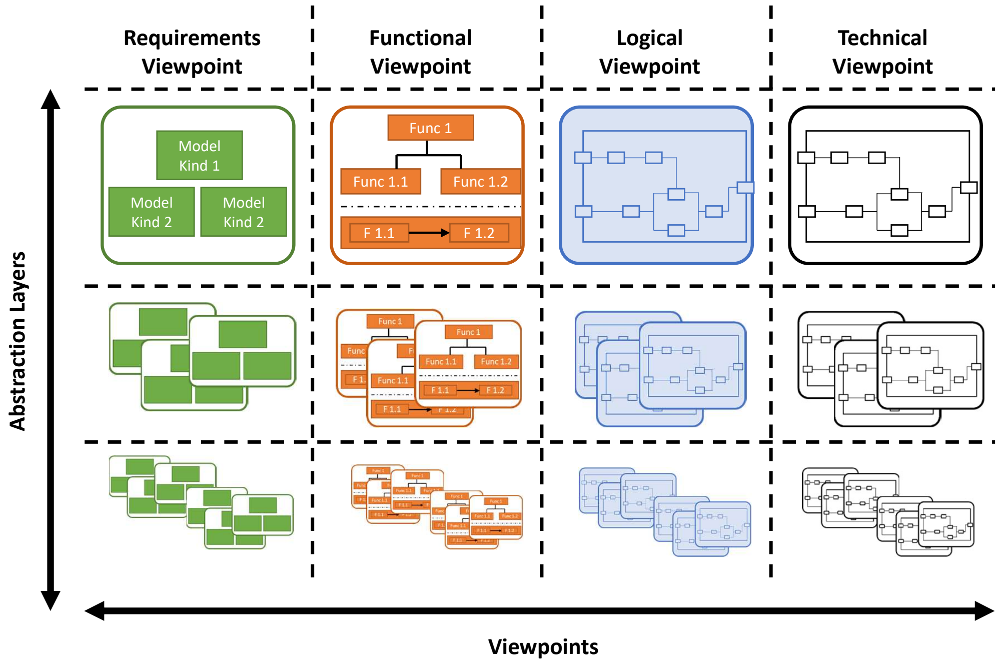


Figure 3.8: Software Platform Embedded Systems (SPES) based on Pohl et al. (2016)

To fulfill the objectives mentioned above, the SPES framework inherits two central concepts, the so-called *Abstraction-Layers* as well as *Views and Viewpoints*. The combination forms a two-dimensional engineering space, as proposed in (Pohl et al., 2012). More precisely, the *Viewpoints* are separated into four different sections in the horizontal axis, which are *Requirements Viewpoint, Functional Viewpoint, Logical Viewpoint* and *Technical Viewpoint*. In contrast, different templates or methods realize each viewpoint. However, the vertical alignment introduces the different abstraction layers according to the *divide and conquer* principle.

3.3.3 Comparison of RAMI 4.0 to IIRA and SPES

Similarly to the previously mentioned reference architectures, IIRA also introduces different viewpoints to describe an industrial system (Lin et al., 2015). Those are divided into four different perspectives: The Business Viewpoint addresses the stakeholders, their interests, or the regulatory context of the system. The Usage Viewpoint defines how the system is expected to be used, usually in the form of activities or capabilities. The Function Viewpoint focuses on the functions of the system as well as their interactions, interfaces, and relations. Finally, the Implementation Viewpoint deals with used technologies to implement the previously defined functional components. Also, the IIRA framework has a shortcoming: In comparison to RAMI 4.0, only four viewpoints are used due to their original purpose, which does not allow a comprehensive description of the system. For example, the implementation viewpoint should specify the technical components that are implementing the functions, their information exchange, and their communication infrastructure. Those aspects are split into separate viewpoints within RAMI 4.0.

When comparing the three mentioned reference architectures with each other, suitable evaluation parameters need to be introduced. Those evaluation parameters are represented with requirements for flexible production systems, as those need to be considered and even fulfilled by the resulting modeling framework. Thus, Table 3.1 indicates how each architecture can fulfill the requirement.

Table 3.1: Comparison of frameworks towards evaluation parameters

Challenge	RAMI 4.0	SPES	IIRA
Multi-discipline collaboration	x	x	
Modeling methodology			
Consistent system representation	x		x

In more detail, RAMI 4.0 supports multi-discipline collaboration by providing different layers and a three-dimensional structure, which addresses most industrial system stakeholders. Additionally, an industrial system is consistently represented when retaining theoretical concepts. On the other hand, SPES also enables multi-discipline collaboration by targeting different domains. However, no consistent system representation is ensured due to the possibility of being extended in-depth. Finally, IIRA allows such an unambiguous representation by being limited in viewpoints. However, due to the shortcoming, a system can be developed on the four-dimensional structure, which impedes multi-discipline collaboration. As far as an applicable modeling methodology is concerned, neither of the reference architectures provides a ready-to-use framework.

All in all, it could be claimed that RAMI 4.0 is considered to be the most promising framework to fulfill the mentioned challenges. While this reference architecture already addresses multi-discipline collaboration and a consistent system representation within its definition, a suitable modeling framework must be provided.

3.4 AutomationML

Several German companies have developed AutomationML to enhance the data exchange between manufacturing engineering tools (Drath, 2021). This standard introduces an object-based arrangement of plant components structured within respective granularity levels. It allows the decomposition into single elements or complete manufacturing cells. To do so, AutomationML uses the computer-aided engineering exchange (CAEX) data format, which is based on extensible markup language (XML) and thus arranges the information accordingly. AutomationML aims to interconnect engineering tools and disciplines by storing all engineering information following the object-oriented paradigm (Lüder et al., 2017).

Originally, AutomationML was developed to exchange data bilaterally between engineering disciplines and in the area of MDE, as seen in Figure 3.9. As Winkler et al. (2016) explains, the results of different engineering phases within a sequential engineering process could be stored and transferred to each discipline with AutomationML. For example, the models created during the system design phase elaborating the plant topology, the mechanical system design, or electrical plans could be stored with AutomationML. The same counts for system models used during the system construction implementation or operation. Additionally, all test plans and specifications could be added to this single point of truth. Furthermore, AutomationML addresses defects and changes within such a MDE environment accordingly, as those are critical factors for various stakeholders. As changes in late project phases are often costly and result in high rework effort, AutomationML also synchronizes this engineering data, and changes management with efficient data exchange approaches. (Winkler et al., 2016)

However, AutomationML provides several innovations compared to traditional approaches (Drath, 2021):

- Meta format instead of data format
- Object-oriented modeling with relations
- Separation of syntax and semantics
- Utilization of existing standards
- Referencing existing semantics
- Modeling of mixed semantics
- Identification of semantics
- Explicit knowledge about unknowns

- Free international standard (IEC 62614)
- Model sustainability
- Different levels of abstraction

A significant advantage can be seen with the object-based arrangement of plant components and their structuring based on CAEX within this standard. This allows to describe objects from a higher-level perspective and complete manufacturing cells up to single decomposed elements on a lower granularity level. Thereby, the single objects could be derived from abstract classes, while the hierarchical structure enables the definition of sub-elements via composition or aggregation. To store this information accordingly, AutomationML introduces four significant concepts of differentiating object-based components within a flexible production system.

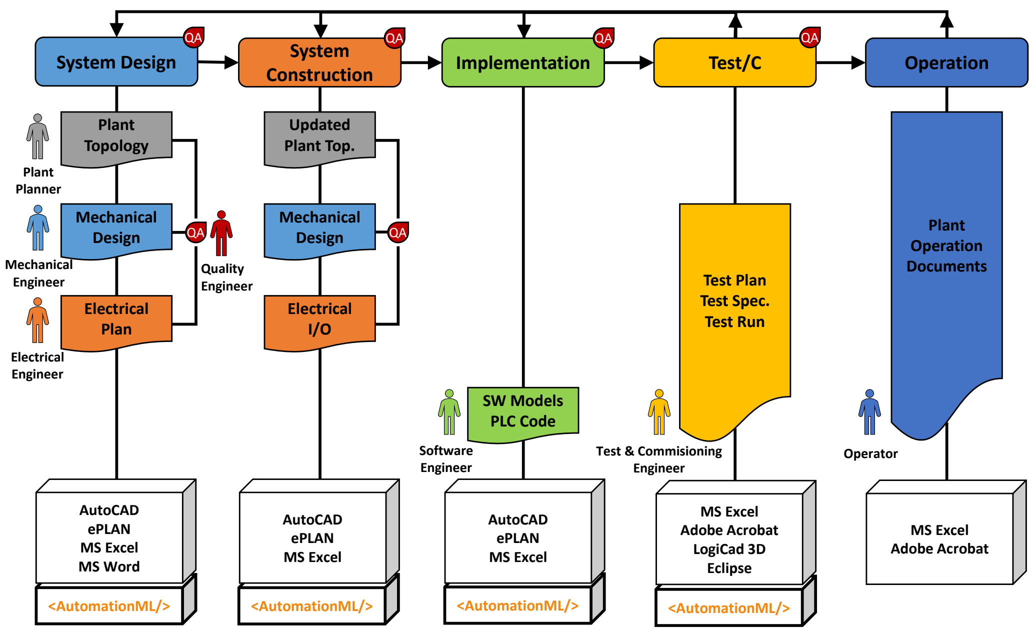


Figure 3.9: Sequential engineering process with parallel engineering activities supported by AutomationML based on Winkler et al. (2016)

At first, RoleClasses describe the abstract system architecture regardless of its technical implementation and thus deals as a foundation for other objects. The semantics is associated with the system elements with the help of this class. This offers the opportunity to describe the meaning of an object in an abstract and manufacturer-independent way. Typing of roles is hierarchically done in so-called libraries. Each role class should be named uniquely within the role tree of the library and might have attributes or interfaces. These attributes and interfaces shall enable an importer of an engineering tool to interpret and process incoming information correctly. (AutomationML consortium, 2014; Drath, 2021; Lüder and Schmidt, 2017)

The so-called InterfaceClass specifies all interfaces and data exchange standards within the industrial system. According to the use case and needs for data exchange, new interface classes could be added. However, each interface class shall have a unique name within the interface class tree of an interface class library to be referenced uniquely. Additionally, each interface class may have attributes. These attributes have to be used and filled with values in each occurrence of an instance of the interface class. (AutomationML consortium, 2014; Drath, 2021; Lüder and Schmidt, 2017)

Next, the SystemUnitClasses have to be defined based on the available system components within the delimited area or domain, like company-specific libraries. System unit classes can be considered reusable system components or templates for system modeling, depending on the point of view. Usually, they reflect a vendor-dependent library of components or devices or a set of templates used within an engineering tool to structure discipline-dependent model information. Within the AutomationML standard, no basic AutomationML system unit class library is defined. Thus, the definition of system unit class libraries is up to the user of AutomationML. AutomationML only defines some rules for system unit class definition. Each system unit class may have sub-objects of the type InternalElement, attributes, and interfaces representing the structure of the modeled class of objects, its properties, and its possible associations. In addition, each system unit class may also be derived from another system unit class using the RefBaseClassPath attribute. In this case, it inherits all supported role classes, sub-elements, interfaces, and attributes from the parent element. (AutomationML consortium, 2014; Drath, 2021; Lüder and Schmidt, 2017)

Finally, Instance Hierarchies store all information, including instances of system components within a particular project. When assigning objects within Instance Hierarchies, major functions or requirements are added. The individual project objects are modeled in an instance hierarchy as a hierarchy of internal elements referencing both system unit classes they are derived from and role classes defining their semantics and interface objects used to interlink objects among each other or with externally stored information. (AutomationML consortium, 2014; Drath, 2021; Lüder and Schmidt, 2017)

This XML-based concept enables to associate engineering tools and disciplines in the context of Industry 4.0 or CPSs by consistently storing all engineering information within the AutomationML file (Berardinelli et al., 2016). Within their work, the authors enable the modeling and generation of AutomationML-files with the modeling software Enterprise Architect (EA). A particular Plug-In has been developed, which consists of a metamodel and a stereotype, allowing applicants to model different AutomationML-related aspects, such as RoleClasses, InterfaceClass, SystemUnitClasses, and InstanceHierarchies. However, as the primary goal of their approach is to cross-disciplinary engineer a production system with SysML and AutomationML, InstanceHierarchies are considered in this work. This originates from the fact that systems engineering with EA describes applied system components. Thus, new approaches dealing with ensuring class-based modeling need to emerge.

By now, AutomationML has a broader application than its original purpose, as it has been paradigmatic in data exchange logistics for engineering networks by exploiting data integration (Lüder, Pauly, et al., 2019). The authors propose an approach for transporting, transforming, selecting, and aggregating data within an engineering toolchain. By doing so, a specific centralized data logistics architecture has been proposed and substantiated by a particular metamodel. The architecture should consider all engineering phases step by step and thus aim to maintain an integrated, consistent, and complete model of the SoI, which is incrementally enhanced. Thus, the approach of the data logistics toolchain implements different functionalities to support those engineering phases of the system model (Lüder, Pauly, et al., 2019):

- Discipline Integration and Management: This function aims to provide methods for representing the engineering information of all involved disciplines consistently. This also includes the representation of dependencies between this information and its propagation across disciplines.
- Change Management: Model versions and releases are considered within this functionality. The information should be provided and maintained, including information on finished engineering phases and system states.
- Completeness Management: In contrast, all information on messing engineering data is required by engineering activities and ought to be provided.
- Consistency Management: The last function realizes the evaluation of consistency rules, including all sources of failure, which are integrated into the overall engineering data model.

To implement and apply the proposed architecture, a particular metamodel has been introduced in (Lüder, Kirchheim, et al., 2019). The single concepts are described in detail in Figure 3.10.

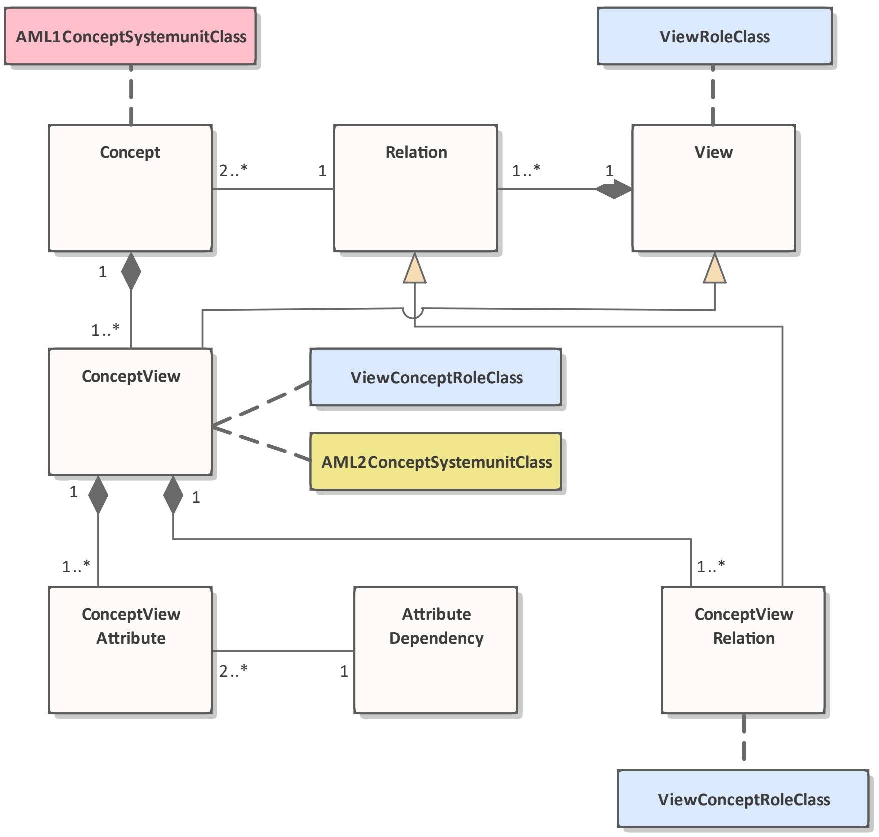


Figure 3.10: Conceptual data integration metamodel with AutomationML based on Lüder, Kirchheim, et al. (2019)

To give further insights, the first step towards AutomationML-based data modeling is identifying relevant views for the various engineering disciplines. To apply these views, model-indicating RoleClasses have been defined. The next step is to identify view-related concepts, which are implemented by utilizing attributes for indicating relevant properties concerning views and interfaces and their relations. Next SystemUnitClasses are developed to create single-discipline data models for each view, while multiple single-discipline data models are combined to form a shared SystemUnitClass library. The resulting SystemUnitClass library might be considered a reference architecture to be instantiated within several system environments. This reference architecture inherits all attributes, models, or views that must be considered in the actual implementation of the system. To enable data modeling within the engineering toolchain, AutomationML appears to be one of the most promising concepts. The statements above give insights into the implementation and, thus, the enabling of the concepts for actual projects. Therefore, using different discipline-related dialects, various engineering disciplines like functional, electrical, or mechanical engineering could be addressed, and respective data models could be combined. Thus, for applying the modeling framework proposed within this thesis throughout the complete modeling toolchain, the concepts of the AutomationML data logistics are used and expanded.

By utilizing AutomationML for the RAMI Toolbox, each of its views might comply with the sub-models originating from MBSE. This means when modeling a system part, like requirements, functional architecture, or technical components, the respective information could be stored within and subsequently exchanged with the AutomationML views. By doing so, the single views are primarily engineered within the basic engineering phase and the help of the RAMI Toolbox and subsequently exported into AutomationML. Then the single disciplines of detailed engineering fall back on this structure and extend them with additional information from each of the disciplines. AutomationML is a central management tool that consistently stores all engineering information in a single model. In conclusion, this means that the engineering models within the RAMI Toolbox need to be synchronized with the AutomationML views to react to changes in one of the models or provide a consistent base for engineering. RTE might be a suitable methodology for dealing with this synchronization.

Additionally, this makes AutomationML a suitable method for model integration and replaces other formats like XML metadata interchange (XMI). Rather than transmitting XML files between the single engineering tools, suitable import or export interfaces allow the utilization of AutomationML as a central model during basic and detailed engineering phases.

Chapter 4

Research Methodology & Case Studies

The research methodology applied in this work should consider the rapid rate of change of available technologies or methods in the IIoT area. In addition, it is advantageous to proceed agilely in such a research project to quickly obtain a possible result and incrementally finalize it within small steps. Furthermore, it is essential to address the problem domain constantly and use already established methodologies.

An appropriate research method must be applied to answer the main research question. As utilizing a particular methodology would be too superficial to be involved in such a complex environment, the research methodology within this thesis uses a combination of multiple methods. The structure and planning process of the chosen methodology is thereby visualized in Figure 4.1 and described in more detail in the remainder of this chapter, which also delineates the used research methodology and evaluation strategy for proposing the holistic industrial systems engineering approach.

As outlined by Hevner and Chatterjee (2010), information systems mainly characterize two basic paradigms to perform research. On the one hand, behavioral science is based initially on methods that originate from natural science, while design science mainly utilizes engineering disciplines. Thus, the behavioral science paradigm tries to analyze what is true, while design science achieves to find practical solutions. Precisely described, behavioral science seeks to develop and justify theories, like principles or laws, that explain organizational or human behaviors and the utilization and management of information systems to support this. On the other hand, design science seeks

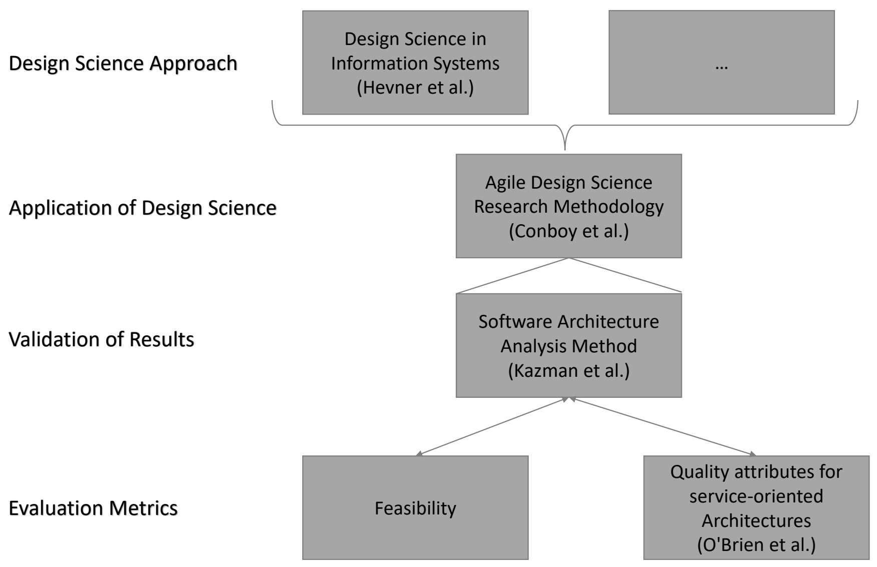

Design Science Approach
Design Science in Information Systems (Hevner et al.)
...
Application of Design Science
Agile Design Science Research Methodology (Conboy et al.)
Validation of Results
Software Architecture Analysis Method (Kazman et al.)
Evaluation Metrics
Feasibility
Quality attributes for service-oriented Architectures (O'Brien et al.)

Figure 4.1: Structure of the applied research process

to create innovations in information systems that are effective and might efficiently be accomplished. In summary, it could be claimed that behavioral science analyzes existing phenomena, whereas design science develops new achievements.

As the goal of this thesis is to develop a modeling framework for industrial systems newly, design science might be the most appropriate research methodology to be applied in this context. The fact substantiates this that design science focuses on solving non-natural phenomena. By doing so, the main goal of design science is to introduce an innovative, goal-oriented artifact representing a technical solution, which solves non-trivial problems, either already existing or emerging in the future.

4.1 Design Science Approach

To consider the rapid rate of change concerning available technologies, the utilized research methodology should support multiple iteration steps rather than providing a solution at once. This means applying an agile method is beneficiary, where a first result can be enhanced incrementally. Additionally, existing and elaborated methods should be used, not reinvent the wheel, and the problem domain's requirements should be addressed. This makes DSR a suitable research methodology for this project. Hevner and Chatterjee (2010) proposed this method to support the development process by

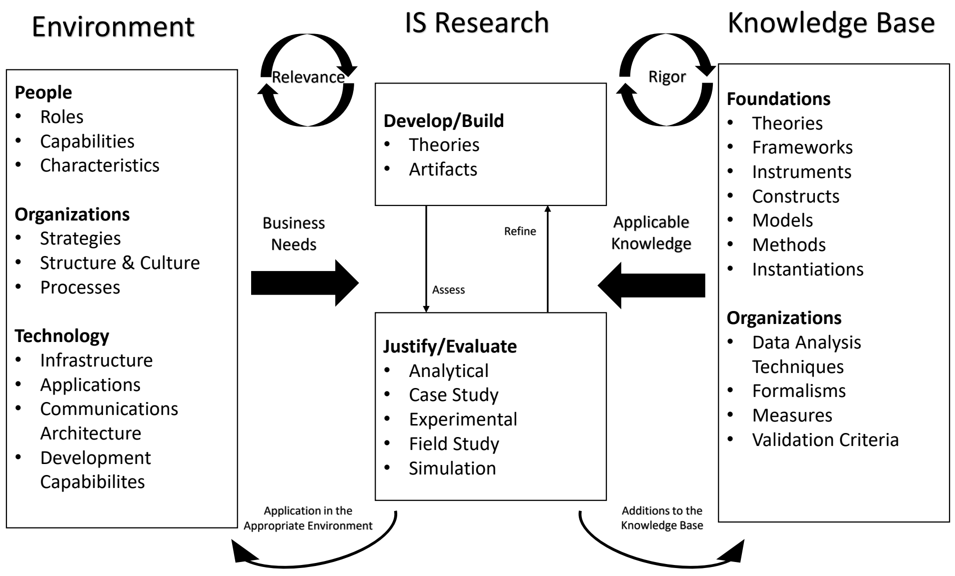


Figure 4.2: Design Science Research in Information Systems based on (Hevner and Chatterjee, 2010)

addressing new or unsolved problems and identifying possibilities to solve them efficiently, as illustrated in Figure 4.2. Those innovative solutions are artifacts or theories that solve current or future problems. The main reason for applying DSR is that the artifact is developed in an iterative way, where new or changing requirements are constantly considered, and novel fundamental theories can be applied. Finally, the resulting artifact is evaluated against requirements or experimentally validated for usability with the help of a prototype or a case study. This makes DSR ideal to be applied for developing a holistic industrial systems engineering approach incrementally. Specifying the framework as research artifacts, its applicability in the system environment can be examined in each iteration and explored theories or methods are added to the knowledge base.

However, the authors of Merwe et al. (2019) proposed several guidelines for conducting design science research in information systems, which are delineated in Table 4.1. The first three guidelines more or less deal with understanding the problem and planning or adjusting the research methodology. Within Guideline 4, the artifact is mentioned for the first time, as it is the main focus of development. Consequently, a suitable method for executing the research study should be selected, and whose results should finally be communicated.

Table 4.1: Guidelines for conducting DSR in information systems (Merwe et al., 2019)

Guideline	Description
Guideline 1:	Contextualize DSR in the field of Information Systems and be able to distinguish between concepts such as design, design science, and DSR.
Guideline 2:	Understand the philosophical underpinning of research and the discourse on the nature of DSR.
Guideline 3:	Obtain a historical perspective of DSR and consult the work of the pioneers in the field.
Guideline 4:	Consider the role of the artifact in DSR and the different views on design theory.
Guideline 5:	Select an appropriate DSR method for the execution of the research study.
Guideline 6:	Strategise how research done in DSR should be communicated in a report such as a thesis.

4.2 Application of Design Science

As DSR is a theoretical approach not specifying a corresponding application method, it has to be extended by an agile, iterative methodology. Thus, in this work, the concepts of DSR in Information Systems deal as a base. In contrast, the so-called agile design science research methodology (ADSRM) is the suitable method for providing an agile iteration process for the research area of engineering sciences (Conboy et al., 2015). This methodology introduces five process steps to be iterated within one research cycle, as shown in Figure 4.3. This allows it to be used in various application scenarios for evolutionarily developing a design artifact or a new theory, like creating a systems engineering approach for flexible industrial systems. The design artifacts represented in this thesis are a domain-specific language for Industry 4.0-based systems, a modeling framework, and the toolchain integration of the basic engineering tool.

The single process steps of ADSRM thereby are:

1. Case Study Specification

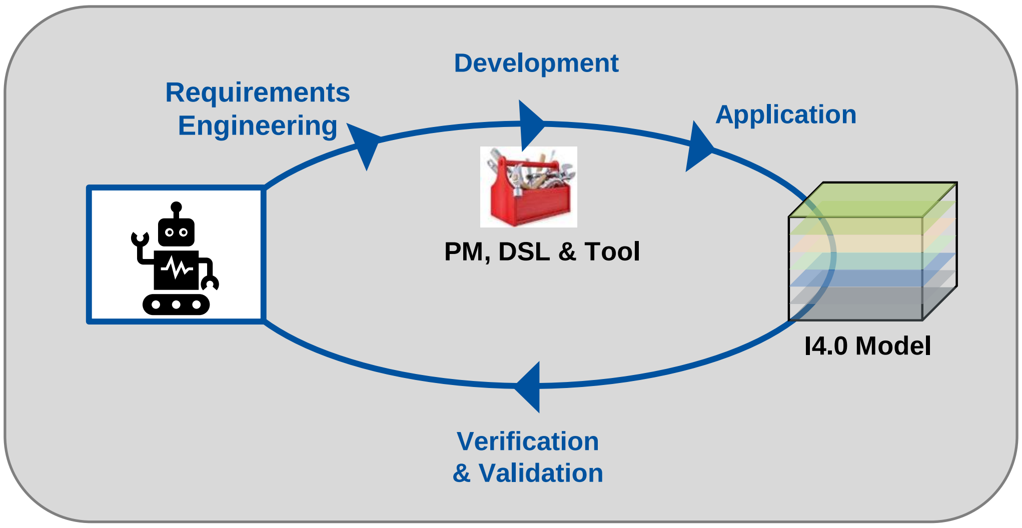


Figure 4.3: Applied Process Steps of ADSRM

2. Requirements Elicitation
3. Implementation
4. Application
5. Verification & Validation

Before iterating these individual process steps, the case study needs to be defined for every single iteration. Based on this case study, requirements for the artifacts to be developed are derived, which are subsequently created and implemented. In this thesis context, the previously mentioned artifacts are specified, and their application is defined and ensured. The last step of ADSRM deals with evaluating and validating the developed artifacts toward feasibility, usability, and applicability. Hence, the validation method mainly uses the suggestions of SAAM (Kazman et al., 1996) to analyze the outcome based on a wide variety of application scenarios. This examines the ability of the described system to fulfill the defined initial quality attributes through its application. To ensure a complete evaluation, results are collected from both the developer and the user context. The results of each iteration are transferred to the next cycle, where an adapted case study or revised requirements offer scope for future developments (Conboy et al., 2015).

4.3 Validation of Results & Evaluation Metrics

As a significant task of the case study is to represent typical industrial scenarios and the consequent derivation of requirements, evaluating the results is also essential. As defined in the analysis method SAAM, the resulting system must be validated against the needs of the stakeholders (Kazman et al., 1996). This is done based on the system architecture, which has been developed to describe the system and its interconnections. The so-called "candidate architecture" should consider static and dynamic aspects of the system and provide a specified semantic interpretation, allowing different models to address various stakeholders. The core of SAAM is to define scenarios for the essential activities of the architecture that need to be supported by the system and its usage. Furthermore, this should address all possible stakeholders, such as end users, customers, system administrators, marketers, and developers, to name a few. The extent to which the architecture addresses the interests of the users or developers connected to the system could be examined based on the applied scenarios. Depending on the system architecture's structural complexity, coupling, and cohesion, the number of interactions between the scenarios and the number of methods should be chosen.

As illustrated in Figure 4.4, the approach of SAAM follows a simple guideline. The definition of different application scenarios to be applied in the evaluation and the development of the architecture to be evaluated is executed separately. Both mentioned elements are subsequently taken and merged within the individual scenario evaluation step. Within this step, the architecture is used to apply within the respective application scenario and based on the specified case study. Finally, the interaction of all scenarios is assessed, and the architecture is evaluated based on this assessment. However, multiple architectures might be compared based on an overall evaluation. Nevertheless, in the context of this thesis, the architecture of the RAMI 4.0 modeling framework is individually evaluated on its own based on three different application scenarios. The evaluation is done in three distinct phases by using three case studies. Subsequently, the interaction between the three scenarios and the application of the case study is assessed, which finally results in the evaluation outcome of SAAM. Thus, the right-hand side of the illustration, which deals with comparing multiple architectures, will not be applied in the context of this thesis.

Another peculiarity of SAAM is that the architecture should be evaluated against quality attributes to validate their intended use qualitatively. In addition to the scenarios, the modeling framework is therefore evaluated against feasibility and substantial

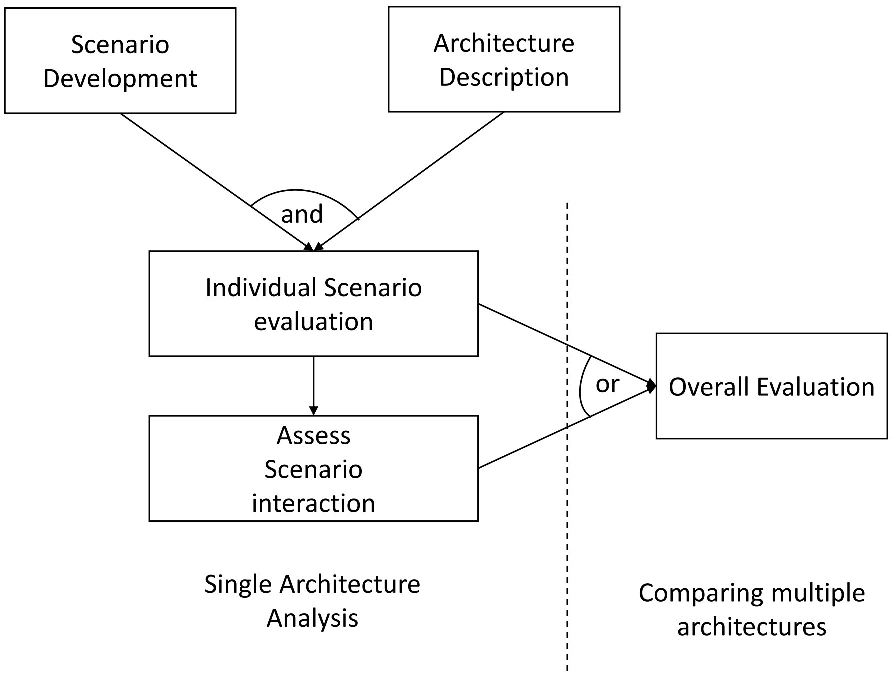

Scenario Development
Architecture Description
and
Individual Scenario evaluation
Assess Scenario interaction
or
Overall Evaluation
Single Architecture Analysis
Comparing multiple architectures

Figure 4.4: Software Architecture Analysis Method based on (Kazman et al., 1996)

system quality attributes for service-oriented architectures (O'Brien et al., 2007), like usability and composability. In more detail, the usability should indicate whether potential stakeholders can use the modeling framework. Regarding feasibility, the focus is on validating the applied methods and technologies f it is meant to develop system architecture with the introduced framework. Finally, the composability should investigate whether the architecture offers a practical solution for any given problem case.

The RAMI 4.0 modeling framework, which this thesis has proposed, aims to be applied in a variety of application scenarios. Those reach from modeling system architectures, exchange of engineering information with other tools, basic engineering of industrial systems, and automation of repetitive modeling tasks. Thus, evaluating all kinds of possible utilization might not be expedient and would exceed the scope of this thesis. Therefore, within the context of this work, three different application scenarios have been specified to be used about SAAM. Those scenarios are aligned with the chapters, which describe the implementation and application of the modeling framework. At first the first method deals with applying the modeling framework for describing industrial system architectures according to a particular methodology and all domain-specific aspects. The second scenario outlines the usage of the modeling framework reference architecture itself. In contrast, the ultimate application scenario addresses

RTE and exchanges to other tools based on the system architecture. The case studies, which are applied in one of the evaluation scenarios to validate the RAMI 4.0 modeling frameworks, are mentioned in detail in the following.

4.4 Applied Case Studies

4.4.1 Metal Profiles for Subway Tracks

The first application example describes the usage of the RAMI Toolbox for modeling metal profiles for subway tracks. The main goal of this use case was to first describe such a subway track according to the specifications of RAMI 4.0 as they are individually produced for each customer. The goal of the modeling process was to automatically define the part thickness and density of the metal plate needed for the subway track. Therefore, a particular regression model is required to be applied.

Specific information like business models or production line infrastructures is thereby provided by a company partner, combined with the desired need for a transformation inheriting the concepts of the fourth industrial revolution. Thus, to integrate Industry 4.0-related aspects, the single subway tracks are produced in sample size 1. Withal, the whole life-cycle of the subway track has to be considered, as well as state-based maintenance or supply-chain challenges. As the system of this case study should be developed based on the specifications of RAMI 4.0, the requirements need to be derived in the next step to fulfill the ADSRM specification as well as provide fundamental directions for creating the piece of software.

The first step to solving this challenge is developing the instantiated system model according to the specifications of RAMI 4.0. There, all production machines, including all sensors and actors, need to be defined and a DSL to describe their needs to be derived. Additionally, production processes are modeled as activities within the Function Layer, while business requirements are placed within the Business Layer. In the bottom Layers of RAMI 4.0, it is needed to add the tagged values to the instantiated system components. Subsequently, the individual subway tracks that need to be produced are modeled. The subway tracks' architecture should include all the information required for producing them with the production system and the corresponding production process. All the necessary information is again stored in the tagged values of the subway track architecture.

This leads to the specification of the following tasks:

- Elaborate a DSL for Industry 4.0-based systems
- Integrate and evaluate the ISO 42010 for refining the architecture of RAMI 4.0
- Follow systems engineering according to a particular development process
- Validate the applicability of the RAMI Toolbox
- Execute a feasibility analysis for future and more sophisticated projects
- Develop functionalities for automating repetitive tasks

Requirements

In this particular scenario, the intention of modeling the case study should consider the following requirements for implementing the case study:

1. 4.4.1.1 Functionality: The system to be developed needs to contain all essential aspects of Industry 4.0 to allow a detailed and complete description. To achieve this, the framework should support the system's creator using well-known methods without raising complexity or administration expenses.
2. 4.4.1.2 Usability: Users may come in contact with Industry 4.0-based systems for the first time. Therefore, usage should be straightforward and supported by demonstration examples and automation tools.
3. 4.4.1.3 Efficiency: After all, the framework is used to increase productivity; this means that resources should be kept low and time-consuming tasks should be avoided.
4. 4.4.1.4 Reliability: The proper creation of a system could be a problem for first-time users. The consideration and prevention of incorrect statements must also be part of the solution.
5. 4.4.1.5 Changeability: RAMI 4.0 and Industry 4.0 consistently change. Therefore, the framework should be adaptable to these changes. In addition, it should be possible to integrate user-specific solutions to react to proprietary implementations.

4.4.2 Siemens Fischertechnik Model

In the context of this thesis, the second applied case study is provided by Siemens and describes the production of plastic housings, as visible in Figure 4.5. The following paragraph describes the original model of the currently installed Fischertechnik production plant. In more detail, this smart factory model should be transformed into a flexible production system. The smart factory has been designed to investigate recent trends and developments in the industrial domain, as outlined by Indri et al. (2018). To do so, some manufacturing stations are set up, each dealing with a different part to install. Moreover, a specific process is executed to manufacture the plastic housing. While this process allows the production of one single plastic housing at once, this could be compared to an original production line. In the following, this process is described in more detail. The original plant consists of a gantry crane with two carriages and four processing stations and a bypass consisting of conveyor belts and turntables. The process is controlled using a Simatic S7-1515-2 PN from Siemens and a decentralized peripheral consisting of a Simatic ET 200 SP module. Moreover, a HMI allows one to select the required workstations and follow the used plastic housing development process. Those workstations represent a milling station, a grinding station, and a specific place for assembly, measurement, and testing. The process is programmed so only this one sequence can be run through, but individual workstations can be skipped. At least one workstation must be selected. Otherwise, the component cannot be transported further. The first carriage transports the components from the infeed conveyor to the workstations. The controller automatically determines the most suitable carriage between the respective stations. After processing, the second carriage lifts the component onto the discharge belt, from where it is transferred to the bypass. The bypass transports the component back onto the system's entry belt or discharges it at the last turntable. In terms of software, the bypass also offers no further possibility of changing the sequence.

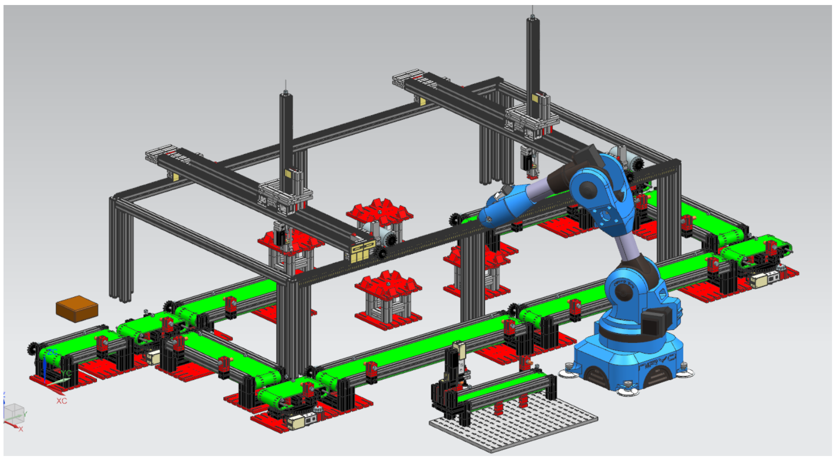


Figure 4.5: Fischertechnik plastic housing case study

Requirements

Requirement 4.4.2.1: As part of the case study, the original smart factory should be transformed to enable additive manufacturing and the configuration and production of individual products. Thus, this already existing smart factory production line should be converted into a flexible manufacturing system according to the possibilities of the IIoT and CPSs. The intended smart factory architecture needs to be described with an architecture modeling framework and developed according to the concepts of MBSE. The following requirements specify the intended flexible production system and are therefore elaborated in the context of this thesis, described in the next. Based on these requirements, the transformation from the original Fischertechnik model towards the model of the flexible production system should take place in the described case study.

Requirement 4.4.2.2: To test and demonstrate the data exchange between different systems, the demonstrator needs to be expanded by a robot and a punching machine, whereby the robot takes over the handling of the components between the punching machine and the bypass. In the future, plastic housings will be manufactured on the system demonstrator. These housings consist of up to three components: a base, a cover, and an insert. There are several variants for each of these components, like a circle or square layout, and a plugged or screwed top, amongst others. Those variants should flexibly be produced, and 3D printed according to the chosen specifications.

Requirement 4.4.2.3: In addition, the content of the plastic housing might be chosen flexibly. The plastic housing's bottom and lid are manufactured in the original manufacturing system. The insert is processed on the punching machine expansion module. The robot deals with controlling and assembling the individual components. In addition, the gantry crane should be able to approach all four workstations in any order. The bypass of the demonstrator is divided into separate modules so that there is the possibility of expanding the bypass at the turntables or connecting other modules or workstations. This means that the parts must be able to be transported to all sides, both on the turntable and the conveyor belts.

4.4.3 Packaging Process

The last of the three case studies applied in this thesis uses a packaging process. This case study is derived from a real-world use case and deals with packing and shipping previously manufactured materials. Thus, it could be dynamically connected to other processes within this area. For example, considering the Fischertechnik use case, the

Table 4.2: Packaging process equipment

Type	Equipment	Quantity
Hardware	Fischertechnik Conveyor Belt 24V	3
	Fischertechnik Vacuum Gripper Robot 24V	1
	Fischertechnik Mechanics 2.0	1
	Fischertechnik Mechanic & Static 2	2
	Fischertechnik Universal 4	1
	Fischertechnik TXT CONTROLLER	1
	Siemens KTP700 Basic	1
Control	Siemens LOGO!Starter	1
	Siemens SIMATIC STEP7 Professional	1
	Siemens Kompakt-CPU1512C-1PN	1
	Siemens SIMATIC OPC UA S7-1500	1
Gadgetry	Isolating Transformer	1
	Switch Cabinet	1
	DIN rail	1
	Cable Material	X
	...	X

packaging case study could be applied directly after manufacturing the plastic housings by using them as packaging material or loading them onto appropriate train wagons. Additionally, several other examples from various other research departments might be able to be applied in this scenario. The packaging case study also interconnects multiple domains, as the loaded wagons must be proceeded within a Smart City or gain energy from the Smart Grid. Due to these reasons, it is considered an ideal use case to demonstrate the applicability of the RAMI 4.0 modeling framework and its interoperability with real-world scenarios.

By doing so, the case study is fully set up from scratch, and no existing environment is used. All hardware, software, and additional material are acquired and installed. At first, the hardware parts are constructed and adequately connected. After that, the associated software to control the single elements is installed. Within this scenario, hardware is also provided by Fischertechnik, while the control equipment is offered by Siemens. In detail, Table 4.2 outlines the used material for setting up the packaging case study.

Based on this material, different scenarios might be covered. Those scenarios build the base for later automation potential and the application of the RAMI 4.0 modeling framework and its interconnection with other manufacturing systems or domains. As seen from the table, three different types of programmable logic controllers (PLCs) are used for executing various tasks, which are finally performed by the Fischertechnik hardware. The remaining gadgetry is used to consider safety requirements or link the hardware parts with each other and connect software to the associated hardware. In more detail, the covered scenario is described:

- The ceiling lighting of the hall should flash if there is a malfunction with the machine. If the light is currently off, it should flash red; if it is on, the flashing should change between white and red.
- Component-specific resource consumption should be determined. By doing so, different kinds of data need to be acquired, like total and individual compressed air consumption as well as power consumption, noise pollution from the machine, or optical component identification by using radio frequency identification (RFID) or barcodes to assign resource consumption.
- All components that leave the Smart Factory should be automatically loaded onto a transport wagon, and the train should be controlled fully. To do this, the train should be stopped so the crane can load. The wagons' correct position must be determined via image recognition. A gantry crane then loads the single wagons.

Thus, the packaging case study is ideal for verifying the automation potential of the modeling framework and interfaces between the architectural model of the system and its actual implementation. This means the packaging process is previously modeled according to the specifications of RAMI 4.0, which is then instantiated, and flexible packing and loading of material are enabled. The model should consider the architecture of the single products that must be shipped and the whole packaging system architecture (Requirement 4.4.3.1). Additionally, the process maps the respective concepts, as each product needs to be loaded individually (Requirement 4.4.3.2). The production system should ensure this flexibility and enable dynamic loading processes (Requirement 4.4.3.3). Each process's needs are exported from the model, and parameters must be set within the PLC. On the other hand, essential production metrics might be collected and flow back into the architectural model. Conclusively, RTE needs to be ensured based on the utilization of this case study, as information might be bidirectionally exchanged via the model and the implemented system. In more detail, an exporting interface might collect all data from the model, which could be engineered system components, process parameters, or filled tagged values, and transfer this information to the implemented system. According to the report, different installations could be done, like setting up new components or adjusting production processes. To keep the system model and implementation consistent, all production data has to be collected and re-imported into the virtual representation of the system.

As this case study additionally interconnects product, process, and production systems, it is a typical example of the definition of a PPR system. The interconnection of all three aspects enables flexible production and consideration of product-to-resource mapping. In more detail, PPR modeling allows investigating the products and their modification by processes executed by the respective resources. In the case of the packaging case study, resources are cranes, assembly lines, and wagons, and products could be anything that needs to be shipped. The process modifies the product, adds packaging material, or changes the location. However, this process might be individually adapted for each product or flexibly adjusted to available production system components. Therefore, the packaging case study might be ideal for investigating the PPR integration of the RAMI 4.0 modeling framework, hence the possibility of modeling product, process, and production system.

4.5 Requirements

This section specifies several user and system requirements for developing and implementing the modeling framework. User requirements are specified toward the intended usability of the framework, while system requirements target the RAMI Toolbox itself. However, the following user requirements have been defined.

4.5.1 Multi-discipline Collaboration and early Verification

The first requirement deals with multi-discipline collaboration and early verification due to multiple stakeholders involved during the engineering process of flexible production systems. When using original document-based approaches, a lot of information is lost, or its availability to other stakeholders is delayed, which led to the proposal of an AutomationML pipeline (Behnert et al., 2021). While validation and verification in the later stages of systems engineering are often costly and require a lot of rework, evaluating the developed results should be done in time. Thus, the RAMI Toolbox needs to provide a central communication point for the collaboration of multiple system architects and ensure early model verification.

4.5.2 Applicable Modeling Methodology

Currently, no suitable modeling methodologies are available for describing flexible production systems. Thus, the second requirement specifies that an application framework needs to be available for system architects within this area. This framework should support them in engineering activities and provide manual and repetitive tasks functionalities. Most important, however, is that traceability within the production is ensured throughout all engineering stages. Thus, the RAMI Toolbox tries to fall back on this requirement.

4.5.3 Consistent and unambiguous System Representation

A consistent and unambiguous system representation is needed to ensure integrity within the flexible production system. Therefore, the RAMI Toolbox should implement unambiguous viewpoints and model kinds, comprehensively describing the system architecture on the one hand and ensuring consistency of modeled system aspects on the other hand. This requirement is considered by providing automatic model-to-model transformation and integrating model-checking functionalities within the RAMI Toolbox.

System Requirements

Additionally, the following system requirements are specified in the context of this thesis:

1. Requirement 4.5.4: A ready-to-use methodology should be provided to enable architecture modeling of flexible production systems.
2. Requirement 4.5.5: As MBSE provides a solution for all three challenges, the methodology should follow these principles when modeling a flexible production system.
3. Requirement 4.5.6: The resulting framework should be easy to install and available for interested practitioners. Moreover, the hurdle to using the framework should be minimized.
4. Requirement 4.5.7: A system architecture developed with the proposed framework should ensure traceability between the architectural elements and be consistent.

Chapter 5

Developing domain-specific concepts for flexible production systems

This chapter deals with the development of domain-specific concepts for flexible production systems. In more detail, the theoretically provided concepts are analyzed towards their practical application. Subsequently, their utilization is ensured by enhancing those theoretical concepts and creating additional value. Thus, this chapter is split into three major parts. The first part deals with implementing a DSL for RAMI 4.0. By doing so, typical notations for industrial systems are utilized, and users are provided with symbols they might easily understand. Additionally, the domain-specific elements are embedded within a metamodel and made applicable via an Enterprise Architect specific model-driven generation (MDG). Subsequently, the respective RAMI 4.0 layers are addressed in more detail after developing this metamodel with all domain-specific elements. This means theoretically described aspects are taken for use, and their application is ensured. Thereby, domain-specific viewpoints and model kinds have been elaborated for each layer, providing users with a suitable environment to model all concerns. Finally, a particular process model is introduced to investigate further the interconnection between the RAMI 4.0 layers, their models, and their elements. This process model guides potential users and ensures a chronological utilization of the proposed concepts.

5.1 RAMI 4.0 DSL

The first aspect of using RAMI 4.0 is the specification of a DSL, including all notations to model flexible production systems. However, as assumed from its name, metamodels are models which build the base for designing modeling languages. To create a DSL, it is essential to understand the application domain, such as the physical world of Industry 4.0 and CPS. Resulting of this, the behavior and context of the physical domain could be analyzed to create a real-world model. The semantics and structure of this model help define the abstractions of the metamodel, and dependencies between the physical and virtual world formulate the connections of its elements, according to Mezhuyev and Samet (2013).

The metamodel representing RAMI 4.0 is composed of a conceptual architecture, constituted of the UML and partly of SysML, as visible in Figure 5.1. It describes the conceptional aspects a language needs to contain to model a system based on Industry 4.0. By doing so, the metamodel is structured in the six layers of RAMI 4.0. On each layer, design elements for describing a viewpoint on a system are provided. The Business Layer, therefore, consists of elements like business actors, business goals, and business cases for representing the cooperation between two actors. With these elements, the desires of stakeholders can be formulated. High-level use cases are specified to realize business cases on the Function Layer to fulfill the defined requirements. Information objects, characterized by a specific data model standard, and the connection paths they are exchanged over are being modeled in the lower layers. The Integration Layer represents the asset administration shell (AAS), a model of the digital twin every physical asset has. Those assets themselves are depicted in the same called Asset Layer.

As the metamodel is a graphical representation of domain-specific elements and their interconnections, a language is designed for a detailed description of those. Similar to the concepts presented in (Dänekas et al., 2014), the conceptual architecture serves as a base to create a specific DSL. This language must be utilized throughout the development process, from designing the system followed by describing it to modeling it. Consisting of an UML profile, the DSL itself can be designed using well-known methods provided by UML. The profile contains all elements previously elaborated from the physical world. Given by UML, the elements consist of a stereotype and a metaclass. The metaclass represents the underlying model element, whereas the stereotype describes the element as it will be used in the DSL. Therefore, all characteristics and attributes of every physical asset have to find their place in the stereotype. By doing

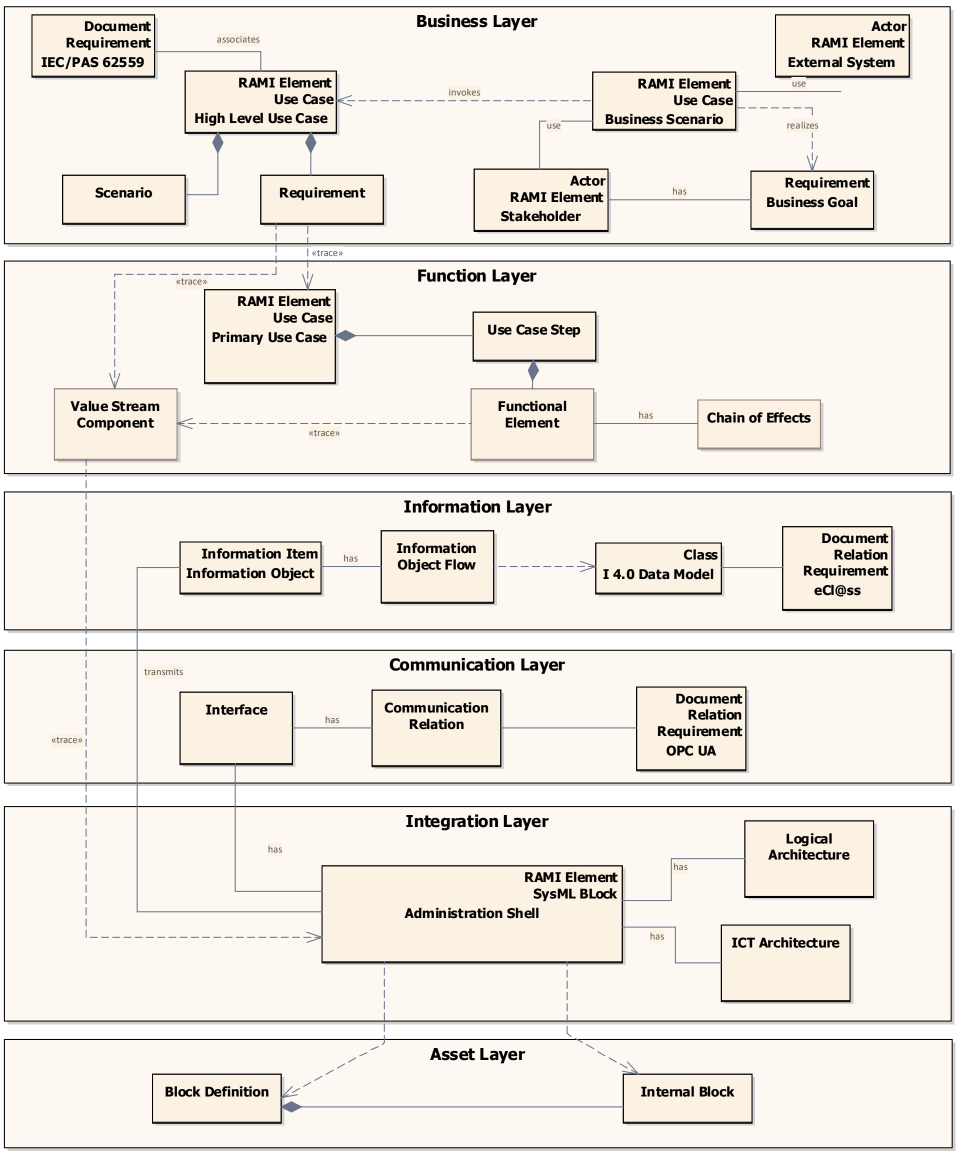


Figure 5.1: RAMI 4.0 metamodel

this, all stereotypes are derived from the general stereotype RAMI 4.0 Element, where global information finds its place. In Figure 5.2, an overview of all domain-specific elements used for modeling system architectures with the RAMI Toolbox is given. The arrangement follows the layers of RAMI 4.0, from top to bottom. The main intention of the image is to show the general traceability between the available modeling elements. However, those are not explained in detail, as they are mainly applied within Chapter 8.

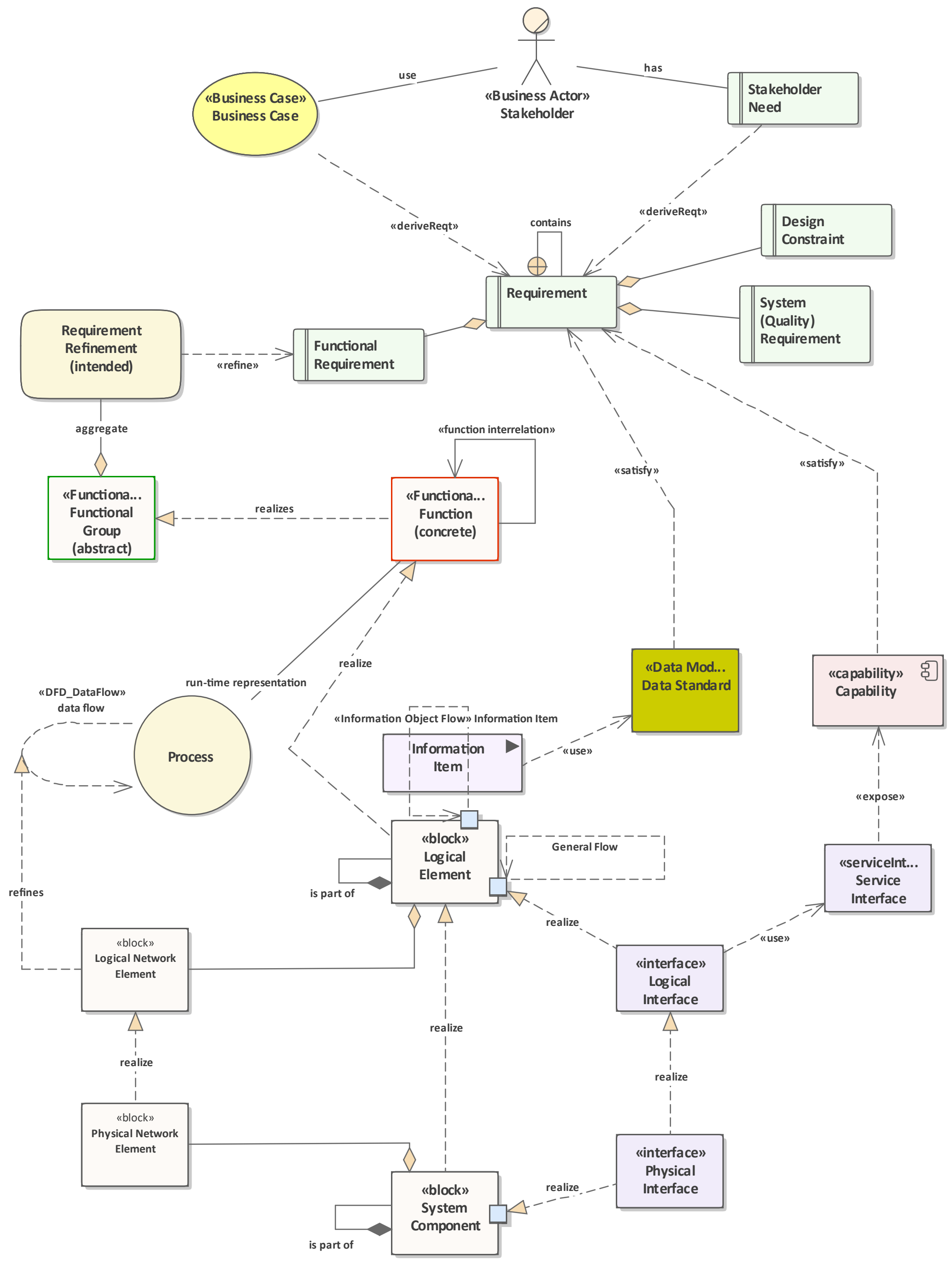


Figure 5.2: RAMI 4.0 modeling elements

5.1.1 Implementation as MDG

There are several software applications on the market tailored to systems development. Concerning its functionality to extend, the modeling tool EA developed by Sparx

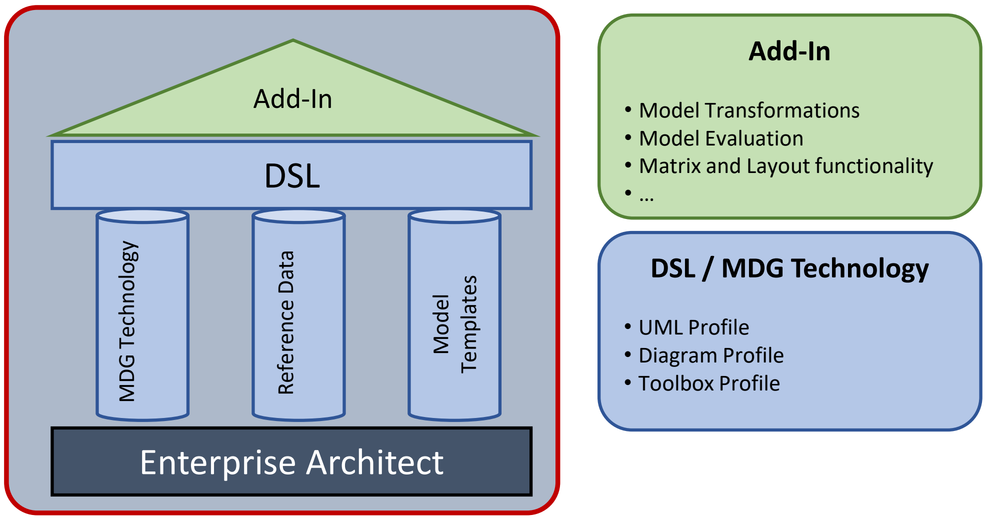


Figure 5.3: EA Add-In for RAMI 4.0

Systems (Sparks, 2009) is suitable for providing an environment to build Industry 4.0-based systems. The already given general modeling functionalities must be extended by implementing the DSL. The result is an Add-In for EA, whose structure is visualized in Figure 5.3. The central part of this toolbox is the DSL described in the previous section. It consists of the UML profile and two other profiles for utilizing a toolset, as well as a suitable UML diagram to describe an industrial model. Adapted from the SGAM Toolbox, it also provides examples of how to use RAMI 4.0. To use this DSL, EA needs to load it during its start-up process to provide a set of tools supporting the modeling of industrial systems.

By doing this, the EA Add-In consists of three major parts:

- MDG Technology, which contains the specifications stated in the DSL and provides them for usage. The compartments of this MDG file are displayed in detail within the following section.
- Model Templates, which support system engineers by providing a fully modeled example and giving information about specific problems. These exemplary application scenarios are not described any further.
- Reference Data contains information about the matrix used in RAMI 4.0 and ensure integration into the model. This needed additional data is used by the RAMI Toolbox and must not be delineated in detail within this thesis.

5.2 Architecture Refinement of RAMI 4.0 Layers

The following section gives insights about the MDG of the RAMI Toolbox and how the layers of RAMI 4.0 are refined to allow domain-specific systems engineering. As the three-dimensional reference model itself is not defined in detail, it should also help recognize the core concepts described on each layer. The following statements should be considered work-in-progress and are evolutionary extended with novel findings. Thus, these specifications have exemplary character and are not yet finalized.

5.2.1 Business Layer

This section delineates the refinement of the RAMI 4.0 Business Layer to enable Requirements engineering. As solely utilizing the established concepts of the business process model and notation (BPMN) to describe business models in such a complex domain would lead to losing important information for the subsequent Requirement Engineering, a domain-specific approach needs to be elaborated. Thus, the work in (Brankovic et al., 2020) deals with interrelations between specific domain-specific systems engineering (DSSE) frameworks, like RAMI 4.0 and SPES. The outcome yields that the only complete one-to-one mapping refers to mapping the requirement viewpoint of SPES either to the Business Layer or the requirement viewpoints of the examined state-of-the-art modeling frameworks. This mapping serves as a basis for further architectural specifications.

The first step towards achieving this is to create an architectural foundation tailored to the peculiarities coming from the industrial domain. According to ISO 42010, one or more viewpoints can address the respective stakeholders and their concerns. As this thesis aims to refine further the Business Layer of RAMI 4.0, the economic aspects of the SoI need to be considered during the system analysis executed on this layer. Therefore, different concerns are taken into account for defining the viewpoints. One of those concerns is the interaction with other systems and the resulting inputs into the SoI or desired outputs. Furthermore, the user interaction, as well as the run-time perspective of the system, are of importance. Since the requirements engineer paves the way for further system development, which is, in this case, conducted on the Function Layer, a specific viewpoint must be provided to address the abovementioned aspects. Therefore, the following views have been defined:

- Context View

- Business View
- Process View
- Requirements View

To give a detailed description and to further integrate ISO 42010-related specifications, each view is realized and made applicable by defining model kinds. As business analysis is a difficult task altering from project to project, a set of available tools integrated within a particular development process should be defined rather than a universal approach. Thus, to consider the context of the SoI, SIPOC has been integrated, which is a process for defining supplier inputs or outputs to customers. Furthermore, the system context modeling language provided by Weilkiens (2016) has been included to model the system environment. The Business View is usually applied with the help of a UML use case diagram, where Business Cases, Business Actors, and their interests can be described. At the same time, the intended run-time process is illustrated with the help of the BPMN. However, to address the latter, domain-specific elements enabling the *Makigami*-method (Gaikwad and Kulkarni, 2014) or the so-called value-stream mapping (Rother and Shook, 2003) are also provided. Finally, the Requirements View is modeled using SysML requirement diagrams but is not limited to it.

According to the designated development process, the first step is elaborating on the system context. The SoI is thereby seen as a black-box where information, matter, or energy can be transferred. Following this concept, Business Actors like customers or the product engineering department provide details on how products should be developed while the suppliers deliver the material. This model indicates the system context by showing what is sent to the system and what needs to be delivered. In the next step, the Business Analysis, the interaction with the SoI is shown using a use case diagram. This model indicates previously elaborated Business Actors and other parties interested in the system. Different Business Cases are drawn inside the SoI, meaning the system's separated application domains. The respective Business Actors are connected with the Business Cases to show their interoperability with the whole system or only a single part. Finally, the interests of each Business Actor are added to this diagram. This helps to find similarities or opposites and is a first step toward the requirements definition.

Subsequently, the business process demonstrating how to achieve the single Business Cases is modeled on different abstraction levels with the help of BPMN. Then, according to the Industry 4.0 vision, the respective production units are visited. Modeling this single process can be repeated for each activity according to the desired granularity level. After the black- and white-box perspective of the SoI has been enabled by modeling its in- and outputs and the business case interactions originating from the stakeholders, the actual requirements can be derived. This is done by summarizing similar stakeholder interests directly into one requirement. With the help of a requirements engineer and elicitation techniques, unique requirements can be determined. Furthermore, it has to be decided whether the identified requirements are explicit of industrial nature or whether they are used as a reference point for the interaction with other domains.

5.2.2 Function Layer

The first part of applying ISO 42010 for describing a system architecture is providing one or more viewpoints for each stakeholder concern. Therefore, firstly the stakeholders and their concerns about functional architectures are elaborated. To give a short overview of the results of this process, some examples are shown in the following. The requirements engineer is interested in accurately formalizing them, while the function developer or the process engineer is concerned with the detailed functional description, including in- and outputs. Moreover, the manager's concern is to fulfill the customers' requirements, whereas the network administrator needs an exact specification of all technical components. Summarized, this results in the definition of the following views:

- FAS View
- Black-Box View
- White-Box View
- Actor Mapping View

After defining the stakeholder's concerns and the corresponding views, the already existing DSL for developing systems based on RAMI 4.0 needs to be adapted. This language contains all modeling elements for describing an industrial system on each of the six abstraction layers of RAMI 4.0. Thus, the single elements, derived from the UML, are assembled to build the metamodel, representing the real world by formulating the dependencies between the elements. However, the first step of adapting this UML profile for providing the possibility to develop functional architectures is the definition of model kinds for each viewpoint. Since the method of choice for developing system functions from functional requirements is the FAS method, its corresponding viewpoint must implement all model kinds that contribute to this goal. Therefore, requirements are refined by use cases in use case diagrams. Moreover, the use cases are further described by activity diagrams. Consequently, the elaborated functional requirements, activities, and actions are summarized into Functional Groups in a particularly designated Function Development diagram. This diagram type implements all DSL elements needed for applying the FAS methods like Functional Groups or Functional Elements. Those modeling elements are additionally used to be traced into each other in an Actor Mapping diagram. However, since the FAS related elements are derived from SysML, the generalization needs to be adapted in the UML profile to add all specific attributes used in SysML. In the following viewpoint, the Black-Box Model, the dependencies between the single Functional Elements and their Functional Interfaces are developed. A SysML block definition diagram or the aforementioned Function Development diagram is utilized for this case. Finally, in the White-Box View, the so-called “chain of effects” is modeled with the help of a SysML internal block diagram. Since the objective of this layer is to elaborate the functional requirements, suitable models for defining the system context and stakeholder goals have to be applied. However, describing this in more detail would exceed the scope of this work and will be proposed in another contribution.

To demonstrate the usage of views, a small example is introduced. This example makes use of the following functional requirements, summarized into requirement clusters: (1) produce lightweight parts, (2) monitor bleed air induction, and (3) turbine movement. As observed from this context, those requirements are spread over different granularity levels of the system. The first requirement deals with the production line, and the others with the system to produce itself. More precisely, the SoI is modeled on abstraction layer 1 and beyond, whereas the supersystem is modeled on level 0. The RAMI 4.0 cube is missing specifications because it only provides the formulation of one granularity level.

To provide a puncture through the modeled case study, the requirement group turbine movement is used for further explanation. Thus, in the first viewpoint of the Function Layer, the FAS method is applied for developing Functional Elements from the requirements. To do so, a functional requirement is refined by a primary use case. The activity diagram of this use case depicts the behavior of the requirement by describing the sequence of events, as depicted in figure 5.4. The displayed image indicates the technical description of how to actuate the turbine. The process is triggered by inducting the inlet air into the engine. Hence, the following action aims to compress this air to increase its density. Subsequently, fuel is added to the engine, which is burnt with the help of compressed air. This combustion process releases heat, which increases the fluid’s total energy and, therefore, can actuate the turbine, called output work. However, the side effects of this movement process are the release of exhaust on the one hand and some internal work on the other. Consequently, summarizing those Actions and other ones from activity diagrams not explained, as well as non-functional requirements, results in the definition of Functional Groups. Thus, in this example, the groups *compress air*, *burn fuel*, and *move turbine* result from these associations. Now, those functions can be described in more detail. This is done by modeling it as a black-box in the corresponding viewpoint and supported by the previously described DSL elements.

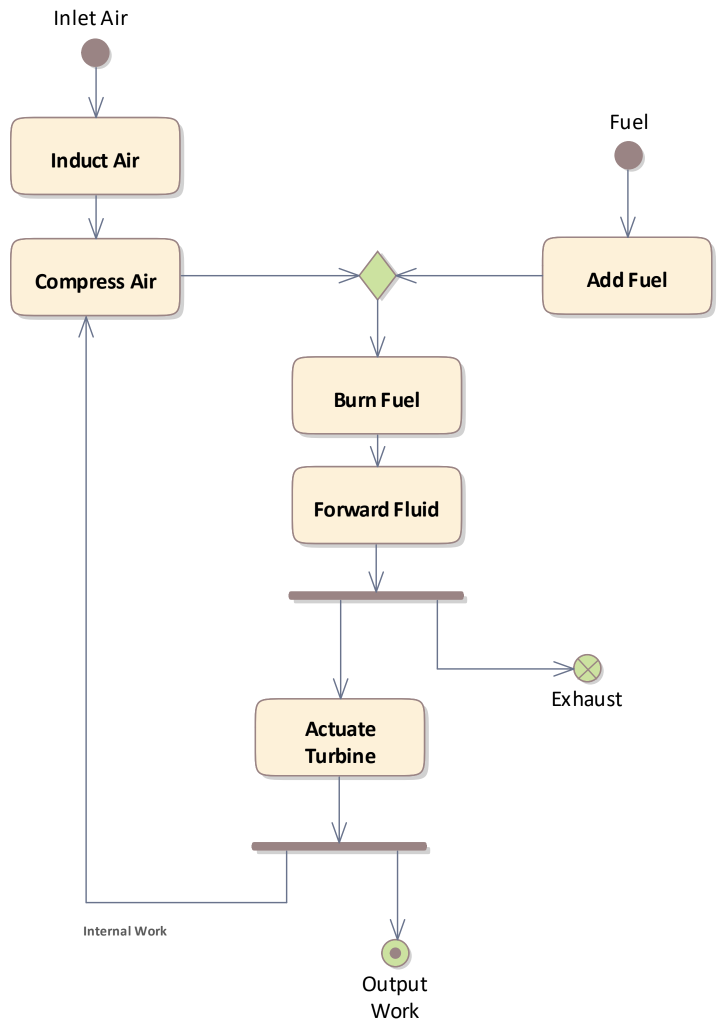
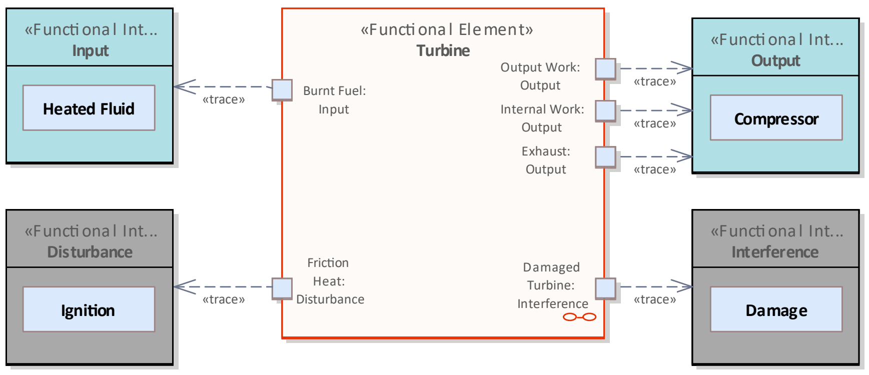


Figure 5.4: Activity diagram delineating the turbine movement

The function is depicted as a SysML block with interfaces defining input, output, disturbance, and interference. Therefore, the input of the function move turbine is the heated fluid, whereas the outputs are internal work, output work, and exhaust. An example of interference would be some mechanical problems, while a disturbance could be too less friction for the turbine to move fluently or too less heat from the burnt fuel. Next, the white-box representation of this SysML block is modeled with an internal block diagram in the White-Box view. More precisely, the hot fluid stream causes the turbine to rotate, turning the output shaft. Furthermore, some energy flows back to move the compressor. In the last step, the exhaust is emitted by particular chambers. The depiction of the black-box and the white-box model is visualized in Figure 5.5.



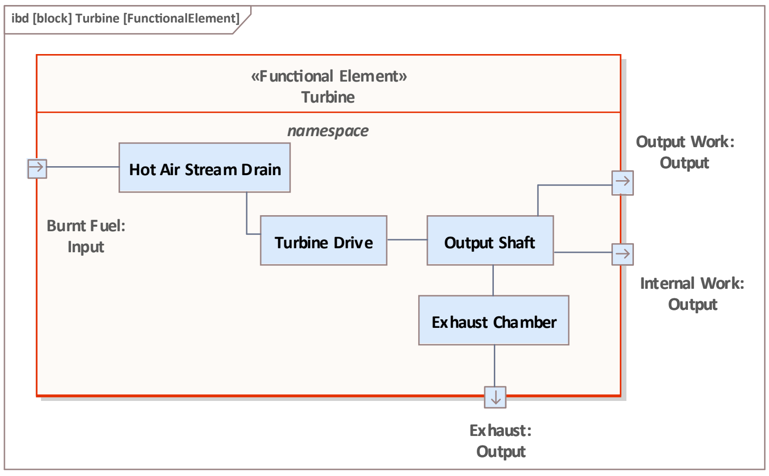


Figure 5.5: Black- and white-box perspective of Functional Element

In the last viewpoint, the Actor Mapping Model, the traceability between the requirements, the Functional Groups, and the Functional Elements are depicted. In this case, the requirement *turbine movement* is traced to the Functional Group *move turbine*, which traces to the Functional Element *turbine*. From this point on, the turbine is considered a part of the system, and the interconnection and technical description can be modeled on the underneath layers of RAMI 4.0. However, the corresponding supersystem can be considered on the top level after modeling the SoI, specifically, the manufacturing parts, on the granularity level 1 and beyond. Since the process for developing this level is the same as mentioned above, only a short overview is given.

Thus, first, the requirements are derived from the ones specified for the single parts of the system, and new ones have been elaborated to create the production line. Subsequently, by applying the FAS method, the single manufacturing processes are described with use cases and activity diagrams to refine the requirements. Consequently, all needed machines, raw materials, production planning, and transport routes can be defined by summarizing the single actions and depicting them as Functional Elements. From this point on, after visualizing the black- and white-box perspective, Industry 4.0 attributes and interconnections can be added to the system components.

5.2.3 Information Layer

The Information Layer of RAMI 4.0 is one of the more important ones, as it allows the utilization of AI or information processing based on the engineered data. This enables system optimization on the one hand or supports operational decision-making on the other hand. Thus, more specific stakeholders and viewpoints have been elaborated for this layer. The result of this refinement, shown in Table 5.1, is described in more detail below. The listed stakeholders and their roles (Ballejos et al., 2008) are taken up, and their concerns in the reference architecture of the Information Layer are more detailed and described in this section.

After the stakeholders and their concerns have been specified for the Information Layer, the next step in ISO 42010 for architecture development is the definition of viewpoints and models to address the concerns. While viewpoints combine several architectural building blocks and manage a larger group of stakeholder concerns, in the simplest case, a single building block is created to satisfy a specific concern. This building block is usually implemented using a concrete model. In conclusion, a model can address one or more concerns, but on the other hand, it does not necessarily have to address a specific concern. It might, for example, only be created as an aid to de-

Table 5.1: RAMI 4.0 and Zachman Framework stakeholder mapping

Zachman	Stakeholder	Role
Planner	Management	responsible
	Regulator	regulator, negative
Owner	Repository Manager	decision-maker
	Auditor	expert
	Repository Operator	operator
Designer	Operational Manager	expert
	System Architect	consultant
Builder	Technology Manager	decision-maker
	Technology Operator	operator
Contractor	Solution Provider	developer
User	Producer/Depositor	operator
	Consumer	beneficiary

scribing the architecture. Since the Zachman Framework already provides elaborated viewpoints, these can be directly applied. The different viewpoints of the system serve the aforementioned stakeholder groups and are arranged accordingly to generate different perspectives on the system. From top to bottom, these deal with the essential aspects of the company, data models at varying levels of granularity, as well as their definition and physical representation. Therefore, the six stakeholders are the basis for Table 5.2. This table is expanded to include the twelve individual stakeholders, and at least one architectural component for fulfilling the concerns is briefly listed in each case.

The first diagram at the contextual level describes all the processes, data, and technologies that are required for the desired execution of the business process, which ensures to fulfillment of the functions of the Function Layer. This way, required data storage or the potential for optimizing the system via information acquisition can be identified. While specific data storage takes place on lower levels of the Information Layer, preprocessing can occur here. For example, the management can decide which information needs to be stored and how or how this information is acquired. In addition, this diagram can be used to identify the automation potential of individual processes. This allows responsible stakeholders to understand the company's required technical architecture and promote its implementation. In addition, data and processes are recognized at an early stage, which means that the individual system components are prioritized by recording requirements.

The logical data model views the system from a lower granularity level. While the

Table 5.2: Viewpoints and Models

Viewpoint	Building Block/Model
System Context	Knowledge Model
	Data Flow Model
	Standard Model
	Constraint Model
Conceptual Model	Data Management Model
	Certification Model
	Database Draft Model
Logical Model	Information Exchange Model
	Data Flow Model
	Inform. System Architecture Model
	Dependency Model
Physical Model	Data Standard Model
	Database System Architecture Model
Data Definition	Database System Design Model
	Information Object Model
	System Maintenance Model
Data Usage	Data Format Model
	Data Set Model
	Data Access Model

system is seen at the highest point of view in the two upper levels, the existing production system is in the foreground. In detail, this means that the core processes of the company, its departments, and the contact with external partners drive the system's conception. This concept is worked out and described from a logical perspective. Therefore, the functional architecture of the production system on granularity level 2 is used here and used for subsequent analyzes in this layer. This defines the exchange of information in the current production system and shows the production processes and the components involved. The specified context and concept aspects are considered, and a design template is given for the physical system development.

The physical data model mainly deals with the technologies used in the system. Therefore, models are required here that define the user data models and database technologies. Before the data models can be created, the needed data standards must first be defined. This allows the technology manager to choose the required standards and pass them on for the implementation or provision of technical solutions. In this case study, the logical data model is used to concretize the relationships between the elements and enrich them with technological aspects. The data standards can be defined for each model of the Information Layer architecture. A separate DSL with elements for the respective standards is provided. This model's result should show which technologies are used in such a system and how they can be represented in the Information Layer. This serves as a template for the subsequent implementation of information architectures and the communication infrastructure within the Communication Layer. In this case study, the dimensions and specifications of the body are passed on to the subsequent production units using JavaScript Object Notation (JSON).

5.2.4 Communication Layer

To further refine the RAMI 4.0 Communication Layer, three different application scenarios are considered:

1. The network developer needs to gain information on how to set up the ICT-infrastructure from the architecture. Additionally, the interconnection between the components and the provided services is essential.
2. The production planner wants to automatically perform tasks on the PLCs, according to previously calculated production parameters.
3. The component provider has to set up the components' interfaces and thus gathers information about provided services, technologies, and exchanged data protocols.

The first step of developing a detailed architecture for the RAMI 4.0 Communication Layer is the specification of views and model kinds to address the corresponding stakeholder concerns. Those artifacts are aligned with the ISO 42010 standard, which is a foundation for the architecture definition. This application scenario introduces three specific stakeholders interested in the Communication Layer architecture. The first stakeholder is described as the network management of the company. The main goals are thereby ensuring the interconnection of the company's single departments or manufacturing units by specifying the ICT network topology. This means the architecture should provide all information about the needed network components and their physical or virtual connection. Subsequently, the second stakeholder is the supervisor of the supervisory control and data acquisition (SCADA). This stakeholder manages the tasks for manufacturing machines based on the SCADA. As one of the main goals of Industry 4.0 is the automation of production processes, OPC UA provides the technical infrastructure to address the SCADA system with suitable protocols and information objects. Therefore, the SCADA supervisor needs to gain information about OPC UA objects within the architecture. Finally, the third stakeholder is the solution provider in general. As each of the system's components is fulfilling a functionality, this function has to be available as a service to other components. Either they can directly execute the function or are provided with the results when submitting a specific input. To address the three mentioned stakeholders as well as their concerns, the following views have been specified within the architecture of the Communication Layer:

- ICT Network Topology View
- Service View
- Communication Standard View

Several model kinds have been defined to ensure the modeling of systems according to the mentioned views. For applying the Interface View, the OMG open-source specification Service-oriented Architecture Modeling Language (SoaML) has been integrated. Additionally, UML and SysML concepts like port or interface diagrams help to understand how the services are provided from the respective components. Regarding the ICT Network Topology View and the Communication Standard View, a particular DSL has been developed to enable modeling within those viewpoints.

After domain-specific stakeholders, concerns, and viewpoints have been defined, the next step is to adapt the already existing DSL for developing industrial systems based on RAMI 4.0. This modeling language contains all elements for describing a system on each abstraction layer of the reference architecture. Thus, modeling elements for developing the Communication Layer architecture must be defined. The metamodel is derived using the concepts of UML. Consequently, two different adjustments have been made to the metamodel of RAMI 4.0. The first adaption is thereby considering the ICT Network Topology View by providing model kinds to contribute to the overall goal of this view. Thus, a deployment diagram has been extended with domain-specific aspects to describe an ICT network architecture. This diagram provides several network components, like mobile devices, routers, switches, firewalls, servers, and cloud-related assets. The toolbox of this diagram offers different network connection types and technologies to interconnect those components. For example, wired or wireless connection types are provided.

The second adaption of the DSL is creating a specific OPC UA client element. This modeling element contains specifications for the identification of the PLC to be interconnected. Via a REST interface, the communication between the client and the PLC is ensured. Additionally, a XML-based template file is added to the RAMI Toolbox. According to the attributes or values entered in the OPC UA element, the XML-file is adjusted. According to the previously modeled specifications, this allows embedding tasks for specific machines within this file. Afterward, the configured file is transmitted via the REST interface to the corresponding PLC, where the embedded task is executed. In conclusion, the mentioned process allows addressing machines directly from the system architecture model, significantly enhancing this area's usability and automation potential.

Different services need to be provided to automatize the Information Layer processes, and the ICT infrastructure needs to be available. In addition, other components are introduced to fulfill the functions defined in the Function Layer, which either provide the service or make use of it. As previously mentioned, the first scenario deals with the Service View of the architecture and how those services are interchanged. This is done by modeling the components via SoaML about domain-specific aspects. Moreover, the data exchange between the services is executed with the help of ports and interfaces. The interfaces and ports are described in more detail at a lower abstraction level so that the solution provider can perform an adequate technical implementation.

5.2.5 Integration Layer

The Integration Layer of RAMI 4.0 is one of the more important concepts, as it interconnects the real-world system and its physical counterpart. Thus, it might be considered the logical architecture of a system described according to the peculiarities of RAMI 4.0. This means logical components are described that can fulfill the function specified in the Function Layer, the information structures defined in the Information Layer, or the services from the Communication Layer. Additionally, real-world objects or physical data is translated into events, which can be virtually processed. The Integration Layer might deal with candidate architectures that describe the system to be operated. This means different solutions are prepared within this level, whereas the best solution is chosen. Thereby, for each solution, a set of related technical concepts is modeled, whose principles are used to support the logical operation of the system. Important considerations are that technical details are not addressed, as those find their place in the Asset Layer. The views to be considered in the Integration Layer are therefore specified as follows:

- Logical View
- Asset Administration View

As far as the Logical View is concerned, different concepts are considered, and a set of solutions is prepared. To fulfill constraints like individual product configurations or reduce lead time to market, companies should enforce the reuse of configurable components and devices to build several variants of their product while minimizing parallel development efforts for products with similar aspects. Therefore, product line engineering has been integrated into contemporary systems engineering. By doing so, different models are proposed, as explained in (Dube, 2021).

Feature & Variant Model

The feature model defines the needed functionality of the product, which is previously elaborated in the Function Layer. To this functionality, a specific variability is added. Thus, this model deals with associating diversity with the needed functionality. Two different kinds of features can be distinguished, mandatory and optional. The Feature model defines all possible values and relationships between the features. An important aspect to consider is alternatives. Either one or the other solution is possible, but combinations of these features should never exist. The single features might additionally be traced to the requirements of the Business Layer. This option allows realizing those requirements for actual system implementations. By doing so, features might also be related to non-functional requirements.

Subsequently, to define the features and their interconnections, product configurations might be specified. This allows the creation of different variants for fulfilling the needed functionality. The best possible solution is picked and refined in the Asset Layer.

150% Architecture

As a result of defining features and variants, the 150% model might be used to represent all the possible combinations of features for all the possible variants. By building various system artifacts, additional models are created. This also includes models that are not needed for specifying a particular system. All possible options are displayed within the 150% architecture and in various models, where a specific alternative might be chosen. Finally, based on this model, a selected variant is desired, resulting in a reduced model. Hence, the resulting system architecture represents selected features and configurable elements.

Another main point of the Integration Layer is the specification of the AAS. While physical components might not be able to communicate in an Industry 4.0 complaint matter, more specific elaborations must be made. As the AAS offers such a framework for locating this additional engineering information, it is promising to be applied for modeling the Integration Layer. However, it is challenging to distinguish between cross-cutting concepts, which need to be taken into account within each of the RAMI 4.0 layers or aspects used to refine technical components further. Examples of cross-cutting concepts or directly assignable concepts are given in Table 5.3.

Table 5.3: RAMI 4.0 cross-cutting concepts vs. assignable concepts

Assignment	Concept
Cross-Cutting	Safety Security Engineering Life-cycle State ...
Function Layer	VDI Functions: Drilling Welding Assembling ...
Communication Layer	Communication
Integration Layer	Identification Configuration Energy Efficiency Condition Monitoring ...

Within the Integration Layer, it is essential to consider aspects like identification, configuration, energy efficiency, and condition monitoring. This means that functionalities that cannot be integrated into physical architectures need to be regarded in the Integration Layer. One of these functionalities is the location of control decisions. While the upper layers of RAMI 4.0 deal with exchanging information via network infrastructures, actual decisions are made within this layer. This means logical elements within the RAMI 4.0 model could be expanded or part of control architectures in the area of IoT. In more detail, such an element might represent a SCADA system or a controller apart from manufacturing system components like machines or transport equipment. Such logical elements hold the information relevant for controlling implementations and allowing decision-making. Examples of such architectures based on the logical elements could be the proposed IoT architecture (Jabraeil Jamali et al., 2020) or a dynamic control architecture based on software-defined networking (SDN) (Bonanni et al., 2021).

5.2.6 Asset Layer

The bottom layer of RAMI 4.0 is defined as Asset Layer. Within this level, all physical system components are elaborated. Therefore, this layer is an interface between basic and detailed engineering disciplines. Resulting of model-driven systems engineering, the system's components, and their interconnections are modeled. The physical system components should represent solutions for previously specified requirements, functions, or logical components. As the foundation for the system implementation, most interfaces between RAMI 4.0 and other engineering tools rely on the Asset Layer's models. To do so, the following views are used to provide suitable architectural concepts:

- Cyber-physical System View
- Physical Connection View

Within the cyber-physical system view, the components of the manufacturing system are described with the help of SysML block definition diagrams and internal block diagrams. This allows for the elaboration of components on various hierarchy levels and enables the possibility of considering composition or decomposition. This engineering information might then be used in other disciplines, like PPR system architectures or exported AutomationML files. Finally, the physical interfaces and the interconnection between those components are modeled within the physical connection view. Thereby, dependencies between these components and technical links could be represented. In contrast to the Communication Layer, where virtual interfaces and services are addressed, the Asset Layer mainly specifies hardware-related interconnections.

5.3 Process Model

As outlined by Kirsch (1996), it is challenging to handle the complexity of distributed systems, especially if they provide critical infrastructures, as is the case for Industry 4.0. A broadly excepted approach to deal with this complexity during the engineering process is the concept of MDE, which serves as an umbrella term for model-based approaches. Derived from this, a standard method for developing systems in specific domains is introduced by Neureiter (2017). The so-called DSSE-Approach is split into three major phases during the development process. The System Analysis defines stakeholders, requirements, and system boundaries in the first step. Following, as a result of the System Architecture, used technologies and components of the system are described. This serves as a base for the detailed Design & Implementation Phase, where the generation of machine-readable code according to the defined model. As previously mentioned, by decomposing the architecture, the DSSE-Approach can be described more thoroughly by integrating the concepts of ISO 15288. According to this, the following phases are decomposed to the technical processes of the standard to describe the development of a system in more detail:

- Business Analysis Process
- Stakeholder Needs Definition Process
- Requirements Analysis Process
- Architectural Design Process
- Design Definition Process
- Implementation

Additionally, every single of these processes includes further engineering tasks for developing a specific part of the system. For example, the requirements specification is part of the Stakeholder Needs Definition Process, whereas the Architectural Design Process includes tasks for developing the information or communication architecture. Each task satisfies a goal, interest, or desire coming from a stakeholder. Therefore, every stakeholder interested in or affected by the system is included during the development of the system and in the architecture description according to ISO 42010.

Considering an Industry 4.0 application as an interdisciplinary SoS, the architecture-focused concepts of MDA appear suitable to analyze, decompose and develop such an industrial system. However, other than the model-driven software development (MDSD) approach, MDA primarily focuses on structuring specifications rather than on the generation of implementation artifacts. By doing so, MDA introduces different viewpoints and their relation, referred to as model transformation. Hence, the previously defined processes deliver the single MDA viewpoints and views, which are described with models. In more detail, the System Analysis Phase, with all its tasks, delivers the CIM, including models for describing business cases, processes, and requirements to give an overview of the system without going into implementation. Consequently, a common understanding of the intended functionality is elaborated during the PIM. The high-level architecture of the system, including superordinate functionalities and generic actors, is composed as a part of this viewpoint. The next step, the System Architecture Phase, deals with used technologies and exchanged information resulting in the PSM, where the system components are described in detail too. Finally, the PSI and its artifacts are delivered by an iterative execution of the Design & Development Phase. The mapping

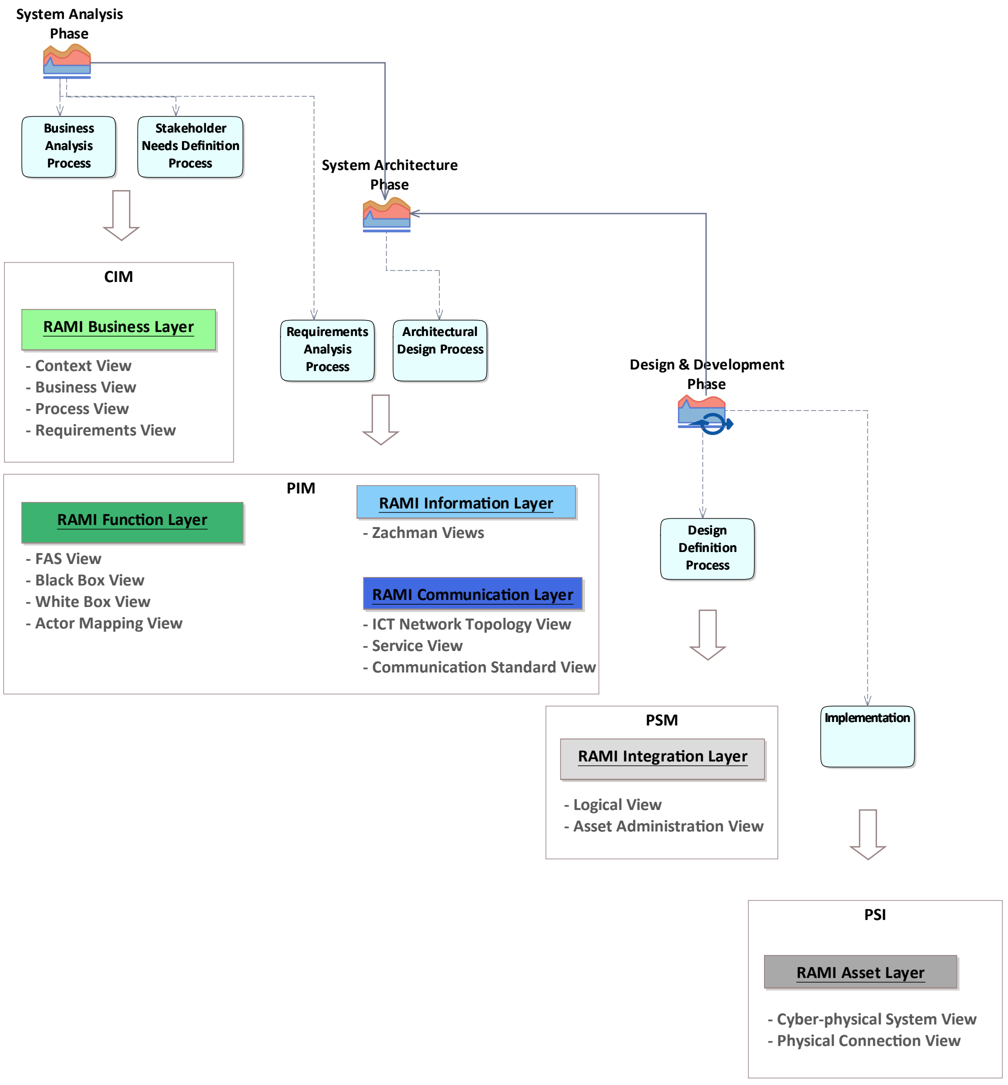


Figure 5.6: Process model for RAMI 4.0

of the individual phases of the DSSE-Approach to the MDA viewpoints are illustrated in Figure 5.6.

Since this process model is developed by utilizing the concepts of the ADSRM, a flexible development method for creating the process model is available, where changes may occur in every process step. However, as this method specifies, a suitable case study must be available to set the boundaries and conditions for future development phases. Therefore, the next step uses the previously defined specifications and requirements for the development. Those requirements are applied to the case study by its practical implementation and evaluated afterward to complete one iteration of the

ADSRM cycle. Hence, the first iteration of this cycle resulted in the creation of a DSL for RAMI 4.0, as previously described. Consequently, the next iteration deals with developing the process model outlined in this contribution. This is initiated by drawing up a new case study and specifying requirements adjusted to the methodology mentioned before. Additionally, the previously developed DSL is used to model the case study according to the development process and enable its evaluation.

5.3.1 Alignment to TOGAF

The interconnection between the two standards must first be analyzed to utilize TOGAF for RAMI 4.0-based architectures. This will enable iterative architecture development based on RAMI 4.0 and further refinement of its architectural model. To do so, the architectures are first compared and mapped to each other to structure the intended approach. Then, the definition of a development process for creating Digital Twin representations of industrial systems needs to be elaborated.

As both frameworks show several similarities, a bilateral mapping could solve the issue of interconnectivity. Thus, in Table 5.4, the result of such a mapping is delineated.

Table 5.4: RAMI 4.0 implementation of TOGAF phases

TOGAF Phase	Implementation
Preliminary	RAMI 4.0 - DIN SPEC 91345
Architecture Vision	RAMI 4.0 Business Layer
Business Architecture	RAMI 4.0 Business Layer
Information Systems Architecture	RAMI 4.0 Function Layer RAMI 4.0 Information Layer RAMI 4.0 Communication Layer RAMI 4.0 Integration Layer
Technology Architecture	RAMI 4.0 Asset Layer
Implementation Governance	AS-IS System Architecture
Change Management	TO-BE System Architecture

While this representation acts as an overview of the mapped architecture modules, a detailed description of how RAMI 4.0-related concepts could solve the respective TOGAF phases are described as follows:

- Prelim: In the preliminary phase, TOGAF wants to identify stakeholders, create architectural views and outline a development process. This is done with the RAMI 4.0 standard itself and the previously mentioned development process by falling back on the criteria of ISO 42010.
- Requirements: The requirements are located in the center of the development cycle and must therefore be considered in each step. Thus, those requirements can be defined in each iteration step of ADSRM.
- Vision: In the architecture vision, it has to be defined why the enterprise architecture is created. This is done in the Business Layer of RAMI 4.0.
- Business Architecture: To support the vision, the business architecture has to be created. Thus, our approach specifies business process, context, and requirement models.
- Information Systems Architectures: Those architectures can be realized with the Function, Information, Communication, and Integration Layer of RAMI 4.0.
- Technology Architecture: Technological system components are derived from the Information Systems Architectures and are located in the Asset Layer of RAMI 4.0.
- Opportunities and Solutions: The selection of technical implementation scenarios based on the previously created architecture is currently made by the systems engineer, but research for automation potential with AutomationML is conducted.
- Migration Planning: The actual implementation of the system is the task of the respective project manager.
- Implementation Governance: Each implementation project has to be assessed and embedded in the Digital Twin representation within RAMI 4.0. To do so, an AS-IS architecture of the deployed system is created.
- Change Management: Based on new business cases or optimization potential, a new system architecture is created, and the development cycle is initiated. This is denoted as the so-called TO-BE architecture of the industrial system.

The TOGAF Architecture Development Method (ADM) describes iterations for three different purposes. First, the development of a comprehensive architecture landscape by repeatedly iterating through the development cycle based on the general purpose of the enterprise architecture. Second, for enabling the creation of the actual system architecture on multiple views to address different stakeholder concerns. Third, the process of constantly evaluating the architecture based on change capabilities (The Open Group, 2022). According to these statements, TOGAF appears to be a suitable methodology to be applied in the context of the RAMI 4.0 modeling framework. Similarities might be recognized by adjusting the three purposes to one of the process models. For example, the reference architecture might be developed and refined within the first phase by iterating through the iteration cycles. Within each cycle, novel findings and adjustments could be integrated. In the context of this thesis, RAMI 4.0 represents the base for the reference architecture to be developed. Based on the outcome of the first purpose, the second purpose allows the creation of existing system architectures. Thereby, views and model kinds are applied that address the stakeholder concerns. This purpose is directly considered within the RAMI 4.0 modeling framework, as compliance with ISO 42010 enables the provision of those concepts. The third purpose describes the constant and evolutionary adjustment of the system architecture, which could be fulfilled by using the greenfield development strategy and the subsequent brownfield development, either by utilizing model-driven engineering or by using digital twin development.

This is why the proposed approach has integrated an AS-IS and TO-BE system architecture development process. By implementing such a process, an iterative development cycle based on RAMI 4.0 can be utilized when evolutionarily developing industrial systems architectures (Calà, 2019). While the AS-IS architecture always represents a mirror image of the currently employed production system, the TO-BE architecture indicates how new digitalization potential or Industry 4.0-related business cases can be implemented within this manufacturing system. A single iteration thereby deals with the execution of one specific business case by adjusting the system architecture based on the RAMI 4.0 cube. However, multiple iterations allow the development of the whole industrial system architecture by continuously integrating all optimization aspects.

Chapter 6

Implementation of the Modeling Framework

After specifying all domain-specific peculiarities the modeling framework needs to consider, this chapter deals with implementing the RAMI Toolbox, including all functions that allow users to describe current or future production systems. By doing so, established standards or methods are first taken for use and adjusted for integration into the modeling framework. This counts for SPES, the Zachman Framework, or the ISO 42010, to mention some examples. After analyzing their suitability for the modeling framework, the actual creation and provision of the RAMI Toolbox are delineated.

6.1 SPES Integration

As depicted in Figure 3.8, SPES creates a two-dimensional engineering space between system viewpoints and abstraction levels. The resulting matrix allows one to classify complex systems and their single compartments. This makes the SPES methodology one of the most promising approaches to developing architectures of industrial systems. Thus, a possible amalgamation between RAMI 4.0 and SPES is investigated within this section.

6.1.1 Mapping RAMI 4.0 to SPES

The first step of interconnecting RAMI 4.0 with SPES has been set in (Brankovic et al., 2020), where architecture frameworks of different domains have been mapped to the SPES framework to enable cross-domain modeling. However, as this approach is targeted to the industrial area, a particular focus on RAMI 4.0 is given in this paragraph. Following this principle, as assumed by their names, mapping the Business Layer to the Requirements Viewpoint is more or less straightforward, as they can be transformed one-on-one. The same method can be applied to the Function Layer by mapping it to the equally called Function Viewpoint of SPES. However, considering the Information and Communication Layer of RAMI 4.0, the SPES matrix does not provide a viewpoint for those. Therefore, the mapping process for interconnecting the two modeling frameworks has to be enhanced. In this case, the two RAMI 4.0 layers are combined and mapped to the Technical Viewpoint of SPES. This leads back to the fact that the interfaces of the single components or the protocols for exchanging data are technical. This means that the mapping is uni-directional since raveling out the information from the Technical Viewpoint would be increasingly challenging. Furthermore, as the Asset Layer also would find its place in this SPES viewpoint, additional information is appended to the technical representation, which hinders the mentioned process even more. At last, as the Integration Layer inherits the Digital Twin representation with all AAS information, it contains the data of several viewpoints. Thus, this RAMI 4.0 layer reaches from the Functional Viewpoint across the Logical Viewpoint up to the Technical Viewpoint of SPES.

Architecture Definition

The first part of applying ISO 42010 for describing a system architecture is providing one or more viewpoints for each stakeholder concern. Therefore, firstly the stakeholders and their concerns about functional architectures are elaborated on in the first place. To give a short overview of the results of this process, some examples are shown in the following. The requirements engineer is interested in accurately formalizing them, while the function developer or the process engineer is concerned with the detailed functional description, including in- and outputs. Moreover, the manager's concern is to fulfill the customers' requirements, whereas the network administrator needs an exact specification of all technical components. As the viewpoints have already been defined in the definition of the SPES framework, the next step is to define model kinds for each viewpoint to realize the architectural view. An overview of the specified model kinds for each viewpoint is summarized.

- Requirements Viewpoint:
    1. Context Model
    2. Goal Model
    3. Scenario Model
    4. Requirements Model
- Function Viewpoint:
    1. FAS Model
    2. Black-Box Model
    3. White-Box Model
- Logical Viewpoint:
    1. Concept Model
- Technical Viewpoint:
    1. Block Definition Model
    2. Internal Block Model

As explained in more detail, the Requirements Viewpoint uses four different model kinds. The Context Model surrounds systems and their in- and output, while the Goal Model deals with identifying the stakeholders and their interests in the system. Both models' outcomes are summarized to identify scenarios in the Scenario Model. However, as the requirements are the most crucial part of fulfilling the equally named viewpoint task, a model has been defined to specify them. Subsequently, in the Function Viewpoint, the first step is elaborating the system functions based on the previously specified requirements. Therefore the FAS methodology is taken for use. The resulting functions are described in more detail in the Black-Box and the White-Box View. In the third column of SPES, the Logical Viewpoint inherits a model where the first realization of the system is delineated. Thus, the Concept Model contains specific elements for describing logical components that fulfill the functions on the one hand as well as digitalizing the mechanical system components according to the IIoT concepts on the other hand. At last, from the Technical Viewpoint, the actual real-world system with all its components is modeled with the help of the block definition diagram or the internal block diagram provided SysML and their corresponding models.

Integration into the RAMI Toolbox

The last step to complete the combination between both approaches is integrating the particularities of SPES into the RAMI Toolbox. Thus, the typical matrix layout of SPES needs to be available for users to structure a system's architecture according to it. An additional step-through user interface has been developed, as seen in Figure 6.1. The respective rows and columns represent the matrix of SPES with its viewpoints and various abstraction levels, while the colors indicate the RAMI 4.0 Layers. According to the specifications in the previous section, a different modeling task has to be fulfilled in each of the squares. For example, in the Requirements Viewpoint, only a certain number of models are available to address the stakeholder concerns. To ensure this, an *Add Model* function appears when one of the squares is clicked. Additionally, an *Allocate* and a *Decompose* function is added in the same step. This enables semi-automatic model transformations, as introduced by MDA. In this case, a further window appears where the trace between the elements can be added to the model.

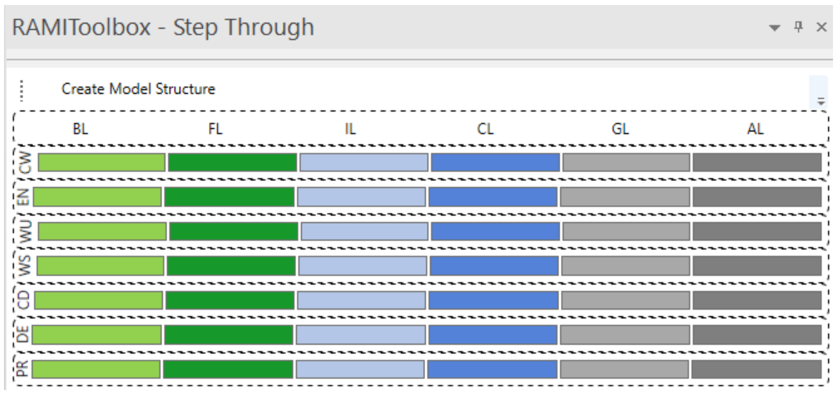


Figure 6.1: SPES window realized in the RAMI Toolbox

However, combining the concepts of SPES with those of RAMI 4.0 inherits multiple problems. As the name of SPES assumes, the viewpoints focus on describing embedded systems. Thus, only four different viewpoints are provided, primarily expressed abstractly. Comparing the Requirements Viewpoint to the Business Layer, the core concept of each layer is the elaboration of requirements. However, while this elaboration is more challenging to be done in an industrial system context than for an embedded system, the Business Layer includes a lot of additional concepts, like business cases, processes, or stakeholders. Within SPES, primarily, the requirements are defined within this viewpoint. As far as the Logical Viewpoint is concerned, specifying the logical architecture and multiple solutions is an essential concept for embedded systems. Doing so could create new product designs, as various product architectures demonstrate different variants or features. Thereby, based on this validation, the best possible solution for the product to be created might be chosen. Nevertheless, as such logical architectures are beneficial when applied to new products designed from scratch, industrial systems mainly consist of already implemented production lines that need to be transformed. Thus, additional viewpoints need to be provided to enable the specification of such system architectures and thereby enable this transformation (Beuche et al., 2016). Overall, it could be affirmed that SPES provides fewer viewpoints to describe all aspects of flexible production systems. The four viewpoints are targeted to model embedded systems architectures, but the six RAMI 4.0 layers better target the industrial domain's peculiarities.

An additional problem is the alignment of each other's architectural characteristics. In more detail, SPES splits a matrix between viewpoints and granularity levels. RAMI 4.0 offers a three-dimensional modeling space for locating all system elements. As adding dimension to SPES would be too difficult and undermines the standardized concept, one dimension from RAMI 4.0 has been removed to align each other layers. As the life-cycle axis is especially targeting products and not comprehensive production systems, it is obvious to remove this axis when describing systems according to both methods. In conclusion, only the RAMI 4.0 layers and the automation pyramid are used to develop industrial system architectures along with SPES. Again, this reduction is more or less counterproductive, as it restricts industrial systems to two domains and does not provide enough engineering possibilities to deal with the full complexity of such kinds of systems. Nevertheless, in terms of developing the proposed modeling framework, this interaction gave the first idea for arranging the panes of the framework.

Finally, the third disadvantage of this interconnection is the missing domain-specific characteristics of the industrial domain. SPES targets multiple domains other than manufacturing, like aerospace or automotive. Thus, most of the used concepts or methods must be kept abstract and applied in various systems. This means the matrix architecture of SPES is too generic to be used explicitly for flexible production systems, as modeling guidelines or engineering tasks are barely specified to work within this context. As RAMI 4.0 provides both suitable viewpoints, engineering guidelines, and domain-specific features, this reference architecture would be ideal for describing such industrial systems. By adjusting those concepts to work with SPES, too many restrictions are set to be adequately applied. Rather than delimiting the needed freedom for the system engineers, the wide-ranging concepts of RAMI 4.0 should be made applicable and supported with proper architecture modeling guidelines. In conclusion, it

Table 6.1: RAMI 4.0 and Zachman Framework viewpoint mapping

RAMI 4.0 layer	Zachman row	Stakeholder
connected world	system context	Planner
enterprise	conceptual model	Owner
work units	logical data model	Designer
station	physical data model	Builder
control device	physical data model	Builder
field device	data definition	Subcontractor
product	data usage	User

could be claimed that the interconnection between RAMI 4.0 is more detrimental than advantageous. Due to this reason, the valuable concepts of SPES should be filtered, investigated, and implemented appropriately into the RAMI 4.0 modeling framework to support the engineering of flexible production systems.

6.2 Zachman Framework Compliance

According to ISO 42010, to create a successful system architecture, it is necessary to know the system's stakeholders to be described and to analyze their interests in the system. This enables the creation of viewpoints and models that satisfy the needs of the stakeholders. In the first step, stakeholders are therefore analyzed from a generic perspective. A wide variety of stakeholders are found who have interests in an industrial system. These are then narrowed down, and those who might have potential interests in the information layer are filtered out. Since the Zachman Framework already contains an established methodology for structuring an information architecture, depicted in Figure 3.7, it is seen as a promising approach to the detailed description of the modeling framework. For this reason, this method's properties are combined with those of RAMI 4.0 to achieve a complete specification. In addition, the Zachman Framework provides ready-to-use viewpoints and their addressed stakeholders. However, these are also kept superficial and must be adapted to the scenario. Therefore, the mapping between RAMI 4.0 and the Zachman Framework can be seen in Table 6.1.

As shown in the table, the layers of RAMI 4.0 and the Zachman Framework can be mapped directly to one another. The connected world is equated with the system context, the work unit with the logical data model, and the field device with the data definition. Only the physical data model of the Zachman Framework can be divided between the two levels of workstation and control device of RAMI 4.0. Additionally, since the Zachman Framework only differentiates between six different stakeholder groups, it is necessary to specify this classification. Therefore, more detailed stakeholders are identified in the next step. For this reason, the proposed stakeholders of Antunes et al. (2011) are used, which imply the roles defined by Ballejos et al. (2008). The stakeholders are described in more detail in the following:

- Management: This stakeholder deals with long-term information system planning. This includes the definition of strategies and goals for the company and its context. Therefore, it is necessary to interact with all business partners involved or the legal environment. This has to be considered, especially the company's data and information. The management, therefore, has a responsible role.
- Regulator: As an external unit, the regulators deal with the corresponding rules for the digital components, such as legal matters or standards. Especially when dealing with information, several aspects must be considered, either about the organization or individual technologies. For this reason, the regulatory role hurts the functionality of the system.
- Repository Manager: This stakeholder plays a responsible and decisive role in making strategic decisions in the administrative area by managing essential resources. This leads to the exchange of information between business partners or corporate entities being administered and ensured in the long term.
- Auditor: The auditors ensure that the company considers and implements the rules and standards to be observed. These are primarily experts who contribute their specialist knowledge in dealing with information.
- Repository Operator: The repository operator serves as the interface between the business decisions and the detailed design of the system. As a result, he designs the blueprint of the information system to be developed based on the operational role.
- Operational Manager: To resolve the conflicts between the technical and business perspectives and to make compromises when creating the information system, operations managers can strike a balance between the respective endpoints. These represent the individual interests of the management and factory employees and can thus act as an interface. This stakeholder, therefore, has an expert role that supports technology decisions.
- System Architect: The system architect develops the actual architecture of the information system and maintains it. In this way, one or the other decides which components are used where and connects all participating entities.

- Technology Manager & Operator: The technology manager decides on the technical means used to ensure system continuity based on advice from the management. In contrast, the technology operator maintains and maintains the infrastructure operationally. For example, the manager, therefore, deals with the different standards and decides on the database system to be selected while the operator introduces it.
- Solution Provider: Through the developing role, this stakeholder is concerned with making all components of the previously defined system architecture available. Either these are developed in-house using company resources, or software or platforms are obtained from other providers. Therefore, this role requires a detailed data format, among other things.
- Producer/Depositor: The producer creates the products intended for the end user. Therefore, this or this must be able to access the different information and structure to use the data for production.
- Consumer: The consumer, on the other hand, uses the result of the information collected to apply it to its interests. This stakeholder, therefore, needs the respective data.

Those stakeholders, which represent the compliance of RAMI 4.0 to the Zachman Framework, are further refined in the following to match the requirements of flexible production systems better.

6.3 Adaption to the ISO 42010

As visible in Figure 2.5, the ISO 42010 provides a conceptual model for interconnecting the core parts of a system architecture. Significantly, the left-hand side of the image explains how stakeholders and their concerns are addressed by viewpoints and in other sequence model kinds. Thus, to define the dependencies within the RAMI Toolbox and other frameworks, this section describes the utilization of ISO 42010 for this end.

6.3.1 Stakeholders & Concerns

To provide an architecture modeling framework for flexible production systems, the first step is to identify relevant stakeholders, their concerns, and the different viewpoints such a system model might have. Finally, the diagrams to model this type of system need to be specified to find suitable models for implementing the views of a flexible production system.

The first step to developing a successful architecture is to elaborate on the stakeholders of a flexible production system. As system stakeholders could change dynamically, specifying a constant stakeholder list is not the right solution. Instead of elaborating on detailed stakeholders, in this work, the number of stakeholder groups is limited, derived from general system stakeholders. This stakeholder list has originally been defined by Antunes et al. (2011). The authors proposed 12 different system stakeholders who are generally interested in any system. As those stakeholders are too general for a flexible production system, more specific stakeholders are introduced in this thesis. Those are represented in Table 6.2.

Table 6.2: General Stakeholders and Industrial Stakeholders

General Stakeholder	Industrial Stakeholder
Executive Management	Executive Management
Regulators	Government
	ISO norms, ...
	Market
Repository Managers	Site Manager
	Head of Product Development
	Head of Production Planning/Manufacturing
Auditors	Quality auditors (e.g., ISO 9000)
Repository Operators	System Designer
	Operational Decision-Maker
	Requirements Manager
Operational Managers	Factory Manager
	Department Manager
	Technology Decision-Maker
System Architects	System Architect
Technology Managers	Provider of Technological Means
	Considerator of Standards
Technology Operators	Maintainer of Technological Means
	Project Manager
Solution Providers	Programmer
	Component Developer
Producers/Depositors	Product Developer
Consumers	Product User
	Product Owner

These interests are defined as so-called concerns within the ISO 42010 standard. To adjust the architecture to those stakeholders, their concerns must be elaborated. As describing every single concern would exceed the scope of this paper, some important ones, as proposed in (Biffl et al., 2017), are listed below:

- Repository Managers like the Head of Department or the Site Managers need to deal with data exchange with external entities and internal processes within their business unit. While technological solutions are not in their primary interest, assuring flawless production processes is one of their primary tasks.
- Technology Operators mainly deal with implementing the required components and choosing the desired technologies.
- The System Architect's main task is to create the architecture of the flexible production system and ensure that the stakeholder concerns are addressed. They chose suitable languages to address them and create the centric model.

6.3.2 Viewpoints & Model Kinds

As previously outlined, it has been shown that both the SPES and the RAMI 4.0 viewpoints are too general to be used for flexible production systems. This indicates that new viewpoints need to be established, which are derived from aligning the architectural peculiarities of industrial systems according to RAMI 4.0 due to their matching with the industrial domain.

As using all three dimensions results in more complexity, reducing the framework to two axes would contribute to a better understanding of the interconnections in flexible production systems. This is underpinned by the fact that the Life Cycle & Value Stream axis mainly targets the value creation possibilities of smart products instead of whole production systems. In conclusion, the other two dimensions of RAMI 4.0, the automation pyramid and the interoperability layers, are used to derive the viewpoints. Their interplay spans a matrix, as shown in Table 6.3.

The table shows that six columns are combined with six rows, resulting in a total of 36 panes. Additionally, it can be seen in the matrix that the product row has been removed, as modeling such a product is a different task than modeling the architecture of the production system. Thus, specifying a modeling framework targeting product development will be done in future work apart from this paper. The interfaces between the production system and the product must be elaborated separately. A particular focus is set on the control device row. There is a discussion that it should be distinguished between the functional unit executing the tasks and the control of this unit (Biffl et al., 2017). However, as the proposed methodology mainly addresses RAMI 4.0, the col-

Table 6.3: Interconnection matrix of hierarchy levels with interoperability layers

	BUS	FUN	INF	COM	INT	ASS
Connected World						
Enterprise						
Work Unit						
Station						
Control Device						
Field Device						

umn Control Device summarizes the functional unit's controlled function, including its intelligence. If a more granular distinction between function and executing unit needs to be done, the framework allows splitting up into more columns. The same counts for the Work Unit row, where an entire work center could be split into single production lines or segments.

As far as viewpoints are concerned, they have been derived from this matrix and the stakeholder concerns. Each viewpoint delivers a particular perspective of the system and might be located within one pane of the matrix or span across multiple panes. By doing so, specific stakeholders might only be located or interested in this part of the system. This means separation of concerns or divide and conquer are two essential principles, as each viewpoint must work on its own at the defined level. In addition, according to ISO 42010, each viewpoint needs at least one model kind to be implemented. Those model kinds can be described with various modeling languages, like UML, SysML, or even a DSL. In particular, the model kinds from the Integration and Asset Layer are used to unite the results from the basic engineering process and provide them to the various detailed engineering disciplines. The elaborated viewpoints and model kinds are listed in Table 6.4.

6.4 Development Strategies

Multiple development strategies or processes are available as the architecture modeling framework for flexible production systems is aligned in the shape of a matrix. More precisely, four different ways are available when creating the system architecture. Every four directions can be targeted starting from any pane without a border. This de-

Table 6.4: Viewpoints & model kinds of the modeling framework

Layer	Viewpoint	Model Kind
Business	System Context	Context & Constraint
	Use Case	Business Case
		High-Level Use Case
	System Usage	Business Process
		Manufacturing Process
	Requirements	Requirement
Function	FAS	Scenario Definition
		Function Grouping
	Function Architecture	Black- & White-Box
	Usable Function	Function Definition
Information	Data Acquisition	Data Exchange
		Business Data Flow
	Data Analysis	Knowledge
	Information Management	Data System
		Database
	Data Usage	Information Object
		Data Access
Communication	Link Analysis	Data Flow
	Communication System	Infrastructure
		Interface & Port
	Usable Services	Service
Integration	Virtual Asset Integration	AutomationML
		Physical Interface
	Digital Twin Representation	DT Architecture
	Usable Production System	ERP
		MES
		SCADA
		PLC
Asset	System Element Class	AutomationML
	Physical Integration	CPS Interface
	Cyber-physical System	CPS Architecture
	CPS Integration	CPS Design

pends on the primary purpose of the architecture utilization. Since developing flexible production systems is a flexible task, no restricted entry points into the modeling task are given. Rather guidelines or propositions are accompanied by this methodology.

First, the distinction between modeling a Greenfield and a Brownfield is essential.

A Greenfield is an entirely new system, while a Brownfield is an already existing one. This distinction must be made because the system can be developed from scratch in a Greenfield, and no existing constraints need to be met. In a Brownfield, the production system is developed based on a current system and aims to enhance it. Both types of systems need to be considered because both types usually occur for industrial systems. With the architecture modeling framework, two kinds of systems can be modeled. By following the proposed system development methodology of TOGAF (Harrison, 2018), this is an iterative process. In more detail, AS-IS architecture describes an industrial system as it is, while TO-BE architecture is an architecture as we want the system to be. This means a Brownfield can be compared with the AS-IS system architecture, while a Greenfield can be compared with the TO-BE system architecture. After defining either the AS-IS or TO-BE architecture of the system, the next iteration of TOGAF deals with a TO-BE architecture under any circumstances.

Thus, three different development strategies are proposed. First, in a Greenfield, a system can be modeled as desired by the stakeholders. This means no previous system architecture is defined, and each pane can be filled from scratch. Second, in a Brownfield, the current system's architecture needs to be modeled first. The matrix layout can describe the currently used production system and locate it within the corresponding panes. The third strategy explains that if either a Greenfield or a Brownfield has been modeled, the next iteration of architecture development is based on an already existing system. With this process, system transformations or adaptions might occur as new requirements or changes to the flexible production system can be specified in the TO-BE architecture. After implementing those changes, the AS-IS architecture is adjusted again.

Another aspect addresses the collaboration across different teams based on the modeled system architecture. While each team is responsible for specific aspects of a component, for example, requirements, functions, or technical aspects, they need to exchange models with each other and build upon their results. Another case would be a change to the existing system model, which should be integrated into all viewpoints and levels of the model. Therefore, different ways to exchange those models or distribute changes from a diagram to other diagrams within the model need to be available based on the matrix of the architecture modeling framework. In conclusion, when modeling a flexible production system according to the architecture, the following four directions can be addressed:

- Model Driven Engineering: From left to right, the RAMI 4.0 layers are iterated top-down, which describes the systems from requirements to technical components.

The method popular with systems engineers is applied, and solutions can be created from previously outlined problem spaces.

- Digital Twin Development: From right to left, the RAMI 4.0 layers are iterated bottom-up, which enables the Digital Twin description of already existing physical elements. This can be applied if the system needs to be defined from existing production lines.
- Component Refinement: From top to down, the context is further refined from factory to workstation up to a single machine. This method might decompose dependencies between requirements, functions, or components into lower granularity levels.
- Factory Integration: From bottom to up, the existing machines can be integrated within a factory. This allows the composition of similar or interconnected system elements to a higher hierarchy level, which is important for analyzing currently installed systems.

Model Driven Engineering

For either Engineering a Greenfield or a Brownfield, the modeling steps are visualized in Table 6.5 as well as in Figure 6.2, which are explained in the remainder of this section. The main difference between the two methods is that within a Greenfield, only the Requirements could be passed to the Function Layer, as no functions are yet defined. However, when engineering a Brownfield, already used functions could be taken to transform either business cases or processes to them. This means a function represents such a business case or process, and they could be traced to each other. Within a Greenfield, those aspects are mainly used to derive requirements, which are then fulfilled in the bottom layers of RAMI 4.0.

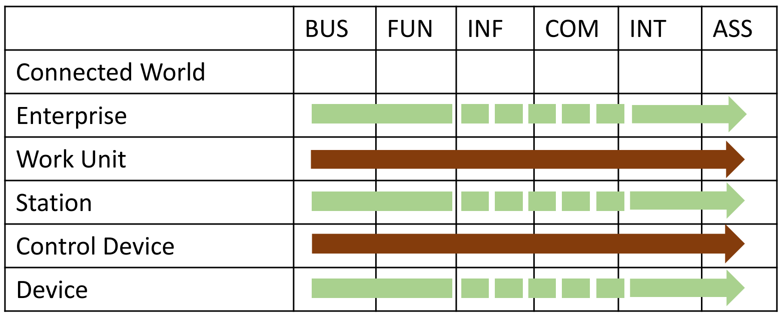


Figure 6.2: Model Driven Engineering - Modeling sequence

Table 6.5: Model Driven Engineering - Business Layer modeling steps

System Type	Modeling Step	Task
Greenfield	System Context	Requirements Elaboration
	Business Case	Requirements Elaboration
	Process	Requirements Elaboration
	Requirements	Pass Requirements to Function Layer
Brownfield	System Context	Requirements Elaboration
	Business Case	Model Transformation to Function Layer
	Process	Model Transformation to Function Layer
	Requirements	Pass Requirements to Function Layer

Subsequently, when making use of MDE, the RAMI 4.0 Function Layer needs to be elaborated, whose modeling steps are visible in Table 6.6. There are main differences to be applied in this layer, whether when engineering a Greenfield or a Brownfield. Within a Greenfield, functions are not yet specified, and no functions exist. Thus, functional requirements are passed from the Business Layer and used for further refinements. Those functional requirements are planned to be fulfilled with the intended system processes. This step is executed with the FAS method, where functions are elaborated based on these requirements. Those functions and their interconnection is then modeled within the functional architecture, while black- and white-box models are used to specify functions themselves. Those steps are also executed within a Brownfield, as the interrelations of already existing functions and their specifications need to be done for existing functions. However, the FAS method is not required for this phase since the functions already exist and might not be developed. Thus, business cases or processes are transformed into the Function Layer, where they are represented with already existing functions.

Table 6.6: Model Driven Engineering - Function Layer modeling steps

System Type	Modeling Step	Task
Greenfield	FAS	Function Development
	Functional Architecture	Model Transformation to Information Layer
	Black- & White-Box	Function Specification
Brownfield	Functional Architecture	Pass Functions to Integration Layer
	Black- & White-Box	Function Specification

After modeling the functions within the Function Layer, the information exchange between the functions is modeled, outlined in Table 6.7. However, this is only done for Greenfield systems, as those system components need to be elaborated. In Brownfield systems, those components already exist and are already inheriting some data exchange. However, when developing new system components, the modeling steps of the Information Layer are executed during MDE. Within this phase, the transformed functions are represented with processes again, which have data input or output. This information is the Information Layer's main focus, while functions could remain as black-boxes. The transmitted data is then passed to the layer below, the Communication Layer.

Table 6.7: Model Driven Engineering - Information Layer modeling steps

System Type	Modeling Step	Task
Greenfield	Data Flow	Pass Data from Flows

As already described in the context of the Information Layer, the Communication Layer is also skipped when engineering a Greenfield. Therefore, the previously defined information exchange must be ensured via stable communication infrastructures, which allow Industry 4.0-compliant data exchange. This communication infrastructure, in more detail, the connection types of passing the information, is the main focus of this engineering phase. Those connections also exist between the respective process and enable data transmission. Thus, solutions for future elements need to consider and realize this information exchange.

Table 6.8: Model Driven Engineering - Communication Layer modeling steps

System Type	Modeling Step	Task
Greenfield	Communication Interconnection	Pass Connection Types

Finally, after specifying all functions, information exchange, and connections, actual system components need to be derived during MDE according to the modeling framework. However, different solutions might be considered before developing real-world system components to fulfill the previously mentioned aspects. Thus, the Integration Layer mainly deals with defining the logical architecture of the system when iterating the RAMI 4.0 layers from top to bottom, which is indicated in Table 6.9. Again, a distinction between a Greenfield and a Brownfield needs to be made. When engineering a Greenfield, the primary goal is to elaborate logical components that can fulfill the required functions and handle desired information exchange or needed communication types. As such, components need to be defined from scratch; usually, different solutions might be possible. This is the point where the 150% solution is taken into account. With this concept, different variants or features could be considered. For example, a crane, a robot, or an assembly line might be applied for transporting goods. Either to the required features or desired functionality, one of the mentioned concepts is more suitable than the others. This decision is based on the modeled variants or component concepts within the Integration Layer. However, when engineering a Brownfield, no different solutions need to be considered, as actual system components are already existing. This means the logical components are directly traced to physical components. In addition, if physical components are already sure to be taken for use, the functions could directly be traced to the Asset Layer, and the Integration Layer might be skipped. Thus, no logical elements need to be defined for guaranteed system components.

Table 6.9: Model Driven Engineering - Integration Layer modeling steps

System Type	Modeling Step	Task
Greenfield	Logical Components Logical Architecture	Model Transformation to Asset Layer Create 150% model
Brownfield	Logical Components	Model Transformation to Asset Layer

Finally, Table 6.10 illustrates the modeling steps when engineering a Greenfield or a Brownfield. As this is the lowermost RAMI 4.0 layer, the MDE process persists at this ultimate phase. This means the used system components are defined within this phase and prepared for implementation or utilization in other system life-cycle phases. At this level, there is no difference between describing a Greenfield or a Brownfield. However, after specifying all physical system components, it has to be continued with digital twin development to ensure integrity within the system model.

Table 6.10: Model Driven Engineering - Asset Layer modeling steps

System Type	Modeling Step	Task
Greenfield	Physical Architecture	Continue with Digital Twin Development
Brownfield	Physical Architecture	Continue with Digital Twin Development

Digital Twin Development

In contrast to MDE, digital twin development deals with digitalizing an already existing system according to the modeling framework. As engineering aims to derive system components by following a specific process, the focus of digital twin development is to enrich the engineering information of system components. The RAMI 4.0 layers are iterated from bottom to top, as seen in Figure 6.3, while each layer adds additional information to physical elements.

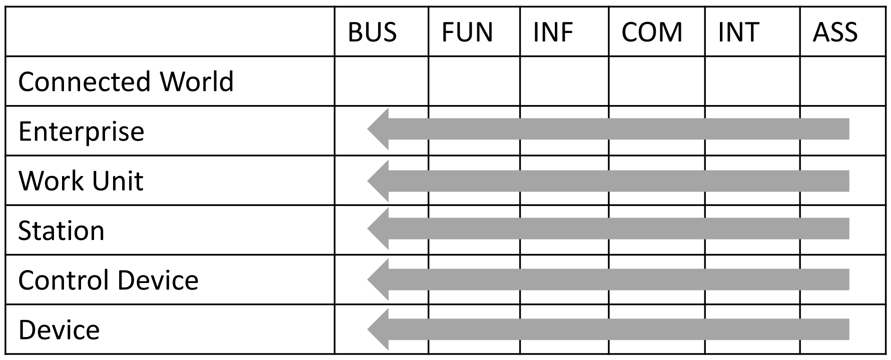


Figure 6.3: Digital Twin Development - Modeling sequence

By doing so, different phases are used, outlined in Table 6.11. However, the single modeling steps remain almost identical for describing a Greenfield or a Brownfield. Thus, the modeling framework's primary purpose is to be mainly applied to engineering tasks rather than describing digital twins. However, by also providing the second named possibility, MDE and digitalization could be executed parallel, which allows exploiting each other advantages. However, by following the phases of the modeling framework, the first step is to model physical elements and their connections. By doing so, it can be considered whether physical interfaces can communicate Industry 4.0-compliant or whether components could actively participate with each other. Those components are transformed into the Integration Layer, which expands this information with additional details. By doing so, each physical's AAS might be modeled to ensure Industry 4.0 capabilities. Thereby, uniform identification, configuration, or energy information is regarded within the model. However, similar to MDE, the logical architecture is used to consider different solutions. The finally elaborated logical components, which result from the logical architecture and are described via the AAS, are passed to the Communication Layer, where Industry 4.0-compliant communication is added. In addition, the Integration Layer also deals with virtualizing data or events created by physical devices. This also includes measurements from sensors or actors, storing this information within databases. Shortly summarized, within digital twin development, the Integration Layer is one of the essential concepts, as real-world assets are virtually processed.

Table 6.11: Digital Twin Development - modeling steps

Layer	Modeling Step	Task
Asset	Physical Elements	Model Transformation to Virtual Components
	Physical Connections	Specification of needed Interfaces
Integration	Asset Administration Shell	Digitalize analogue information
	Logical Architecture	Consider different Solutions
	Logical Components	Pass Logical Components to Communication Layer
Communication	Services	Pass Services to Information Layer
	Communication Infrastructure	Pass Infrastructure to Information Layer
Information	Data Exchange	Model Transformation to Function Layer
	Date Model Standards	Elaborate needed Standards
Function	Functional Architecture	Model Transformation to Business Layer
	Black- & White-Box Functions	Function Specification
Business	Requirements	Trace (non-)functional Requirements

After digitalizing all physical components, interfaces, or events and modeling the AAS of products, the next step is creating Industry 4.0-compliant communication. As the name assumes, this is done within the Communication Layer of RAMI 4.0. However, as the components and their interconnection already exist, other modeling steps are applied than those used in MDE. The primary goal of engineering is to create new components out of requirements, while digital twins expand already existing system compartments. Thus, the Communication Layer defines services provided by the single system components. As RAMI 4.0 is defined as a SoA, elaborating services to provide functions is a significant aspect. Thus, based on existing system architectures, components are extended with providing services, while other components require those. This modeling step is executed within the Communication Layer using provided or required interfaces. A communication infrastructure needs to be modeled to create an infrastructure that can distribute the services over the whole network. This infrastructure consists of networking components like routers, servers, switches, and industrial connection types. Thus, bi-directional exchange will be enabled by developing this aspect of the digital twin.

Subsequently, the information exchange between the components needs to be modeled after creating the services and the communication infrastructure. As each service requires or provides some information, the exchanged data is specified within the Information Layer. By doing so, information objects are utilized, which represent exchanged data collections. Those information objects are directed and realized with so-called information object flows. In more detail, after specifying this model, components might be expanded by data structures, which need to be implemented to enable Industry 4.0-compliant data exchange. Subsequently, data model standards could be defined after modeling the information objects. Those standards could be connected to non-functional requirements and ensure standardized data structures to be utilized, like OPC UA or JSON.

The following modeling steps are located within the Function Layer of RAMI 4.0 and could be compared to the model-driven system development of a Brownfield. These steps solely deal with modeling the functional architecture and refining the functions with black- and white-box models. Nevertheless, after elaborating on all services provided by components and the needed data exchange, this information could directly be traced to the Function Layer, as services are realized by functions and information exchange is implemented by in- or output interfaces of the functions. In more detail, a function could be considered an additional viewpoint to a component, including services and data. However, a function allows a more detailed description of this information and thus enriches components with additional engineering information. To model these details, black- and white-box models are used, which indicate how inputs are transferred into outputs.

Finally, as the last step of digital twin development, the Business Layer specifies the requirements. Since this layer plays a significant role within MDE, as requirements need to be defined within this model phase, within digital twin development, the Business Layer only inherits one single modeling step. Nevertheless, this step is essential, as it traces requirements to the system components. Similar to MDE, functional requirements are interconnected with functions that fulfill these requirements. However, non-functional requirements could be interconnected with multiple digital twin concepts. For example, the communication infrastructure might meet such a non-functional requirement or be constrained. The same could count for data model standards within the Information Layer or configurations and energy efficiency within the Integration Layer. Thus, to ensure traceability between the system components and the requirements, this model step is also essential. However, if single requirements remain without a trace to system components, the currently implemented system cannot fulfill them. This means, from this point on, MDE, as proposed by the modeling framework, could be applied to elaborate new system components so that each requirement can be fulfilled.

Component Refinement

When iterating the modeling framework from top to down, the hierarchy levels of RAMI 4.0 are passed, as seen in Figure 6.4.

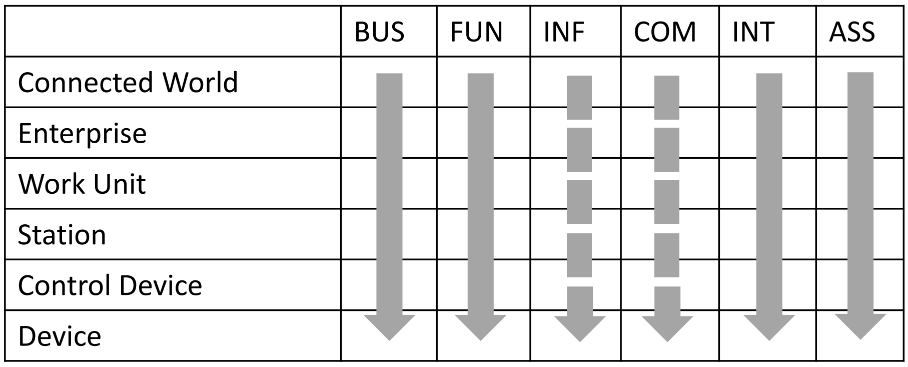


Figure 6.4: Component Refinement - Modeling sequence

Those are derived from the automation pyramid and therefore enable the specification of components on different abstraction levels. This means single components must be aligned according to their functionality or intended purpose.

However, as components might exist of sub-components, the alignment to multiple layers is inevitable. As the SoI might be a complete station on a superior hierarchy level, the used machines are in focus on a lower level. In addition, machine parts are in focus on the next lower level in turn. However, this structuring implies that machine parts or the machine itself are part of the station, while the station exists of multiple machines. Concluding, a hierarchical structure has to be considered when modeling a system with the modeling framework. Hence, Table 6.12 indicates system parts that might be decomposed into sub-parts on lower granularity levels.

It becomes evident that physical components could be refined by falling back to the previous example. This is located within the Asset Layer, where the system ar-

Table 6.12: Component Refinement - RAMI 4.0 models to be refined

Layer	Modeling Step	Task
Business	Requirements	Refinement of Requirements
Function	Functions	Refinement of Functions
Asset	Components	Refinement of Components
(Information)	Data Management	Refinement of Information
(Communication)	Network Infrastructure	Refinement of Technologies

chitecture represents its physical counterpart, the implemented manufacturing system. However, functions might also be decomposed, as higher-level functions include multiple sub-functions to transform inputs into outputs. Those sub-functions are superordinate functions on the granularity level below and contain in- or outputs that need to be addressed. The sub-functions to transform respective inputs into outputs are again modeled at a granularity level below. Additionally, requirements are also able to be fulfilled on multiple levels. An abstract requirement might be more specified on a layer below. For example, considering key performance indicators (KPIs) to be fulfilled, the enterprise level is solely interested in fulfilling any of them. However, on the granularity levels below, it is of importance which KPIs have to be fulfilled and what results need to be achieved to fulfill them.

As far as digital twin development is concerned, the network infrastructure could also be decomposed. While network topologies and connection types are the focus of interest on higher levels, used protocols or technologies to realize the interconnections are modeled at the below levels. Finally, actual interfaces to assure this interconnection are part of the lowest granularity levels. Similar scenarios could be mentioned for the Information Layer, where abstract processes and databases are modeled on top and refined in single database tables or information items.

Factory Integration

The opposite of component refinement could be considered factory integration. In this scenario, no components are refined but summarized into superior components. If similar concepts are recognized, they might be summarized, and a superior system part is specified. This process is visualized in Figure 6.5.

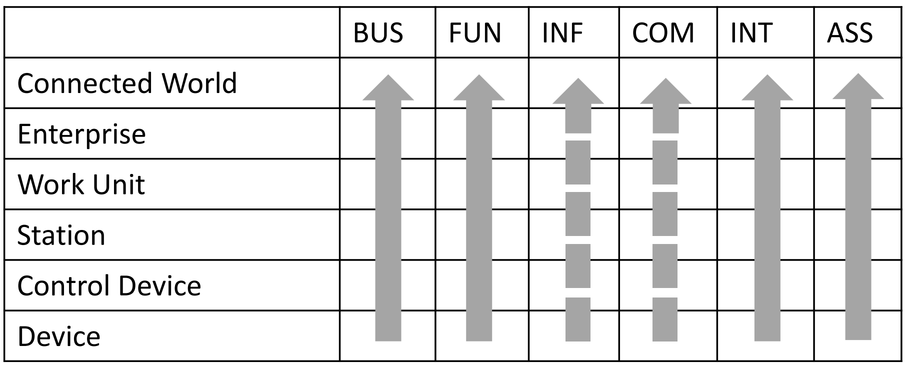


Figure 6.5: Factory Integration - Modeling sequence

Suitable models for executing this step are again shown in Table 6.13. As illustrated, the same modeling steps mentioned within the component refinement are described, leading back to their correlation. However, shortly summarized, similar requirements could be combined to a super-ordinate requirement on the higher granularity level, which is the same for functions or system components. Regarding the Information and Communication Layer, when specifying interfaces or information items, technologies to transmit information via data standards or connection types are determined on superior granularity levels.

Table 6.13: Factory Integration - RAMI models to be summarized

Layer	Modeling Step	Task
Business	Requirements	Summarizing of Requirements
Function	Functions	Summarizing of Functions
Asset	Components	Summarizing of Components
(Information)	Data Management	Summarizing of Information
(Communication)	Network Infrastructure	Summarizing of Technologies

6.5 RAMI Toolbox

For implementing and applying the architecture modeling framework, several software applications currently on the market can be tailored to systems engineering. One of the best-fitting ones is the modeling tool Enterprise Architect, which Sparx Systems have developed. This tool provides an environment aiming to architect and design flexible production systems, as it contains the foundations and possibilities to extend functionalities. Thus, the already given general modeling functionalities must be extended by embedding a DSL to implement the architecture modeling framework. This DSL is implemented with the help of an Add-In and according to the specifications of a MDG, as it is called within EA. Thereby, the core concept of the DSL is the UML metamodel, which represents the conceptual architecture of the modeling framework. By doing so, the metamodel inherits all elements needed to describe the application domain of the flexible production system and shows the dependencies with the real world of a physical manufacturing plant.

For each of the panes within the matrix of the architecture modeling framework, the metamodel provides design elements for describing the corresponding viewpoint of the system. Mainly, these elements consist of diagrams, toolboxes, and modeling elements. According to the particular stakeholder concerns, a diagram originating from UML, SysML, or other languages like SoaML is used. However, in some cases, even a DSL-specific diagram has been developed and provided. The toolbox thereby maps the corresponding modeling elements to the diagrams. This means that each of the architecture stakeholders can express their interests with models and symbols he can easily understand and relate to. For example, requirements engineers are provided with use case or requirement diagrams, while network administrators might use a DSL ICT-network diagram. Technical aspects or functions are thereby considered with SysML block definition or internal block diagrams. To provide this DSL to the stakeholders, the so-called RAMI Toolbox has been developed as Add-In for EA.

This DSL is implemented within the MDG, which represents an XML-file composed of all needed design elements. During the start-up process of EA, this MDG is loaded by the Add-In and subsequently provides a set of tools supporting the architecture modeling of flexible production systems. Additionally, the RAMI Toolbox contains several other elements like modeling templates, reference data, or configurations. The RAMI Toolbox library allows to automation of repetitive tasks and provides a graphical user interface (GUI) that guides the modeling steps or executes model evaluations. Also, event handling with EA is done with the help of the RAMI Toolbox, which strongly enhances the usability of the architecture modeling framework.

In the following, as the main focus of the modeling framework is set towards feasibility, usability, and composability, the application of the RAMI Toolbox is an important focus to be achieved by this particular thesis. Thereby, user interfaces and functions are outlined that help any system engineer within their modeling activities. The main window of the RAMI Toolbox GUI is shown in Figure 6.6. In this image, it can be seen that the framework structures a flexible production system into 42 different panes. Those result from the amalgamation of the RAMI 4.0 interoperability layers and the automation pyramid axis.

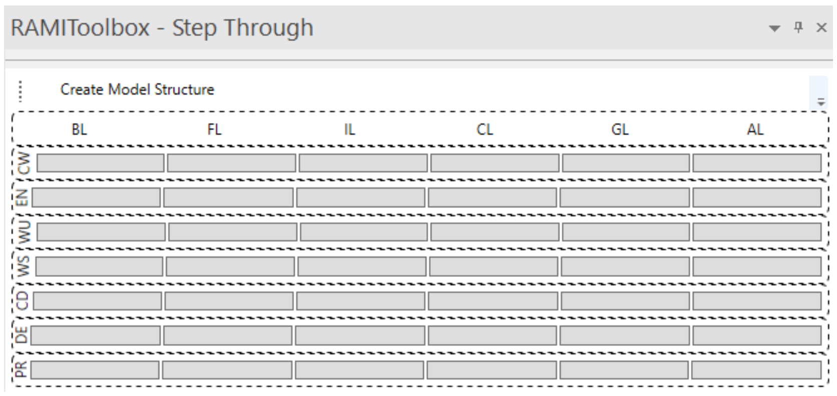
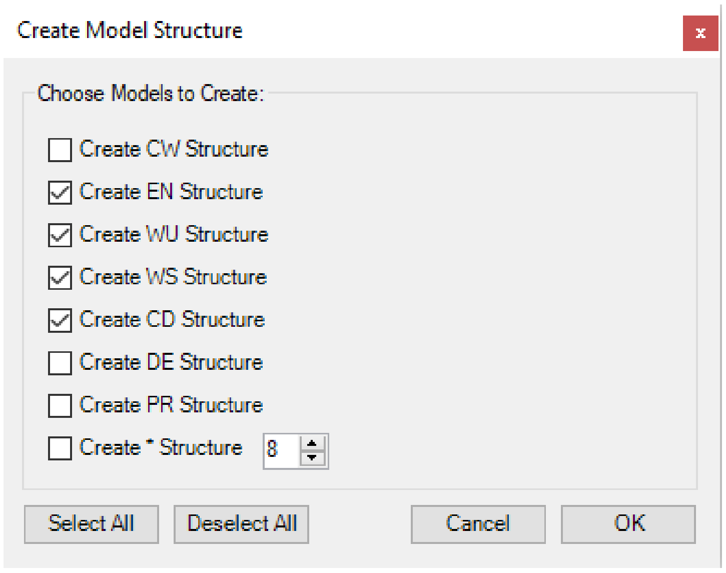


Figure 6.6: Main GUI window

In more detail, the horizontal alignment depicts the layers from Business Layer to Asset Layer. From top to bottom, the domains of the automation pyramid, as defined by RAMI 4.0, are described: Connected World, Enterprise, Work Unit, Station, Control Device, Device, and Product. Those panes span an engineering space in a matrix, where systems can be located according to their characteristics. The granularity could be determined from top to bottom, whereas high-level systems are located at the top, while detailed sub-systems or system components are located at the bottom. All seven hierarchy levels should be fully described for a clear, flexible production system description. Finally, from left to right, the complementary aspects of the system are spanned over the RAMI 4.0 layers. This means requirements and business models belong to the Business Layer, functions to the Function Layer, or interfaces to the Communication Layer.

Above the matrix of the system alignment, a button called “Create Model Structure” is visible. This function helps set up the project structure, including the needed granularity levels within EA, as shown in Figure 6.7.


Create Model Structure

Choose Models to Create:

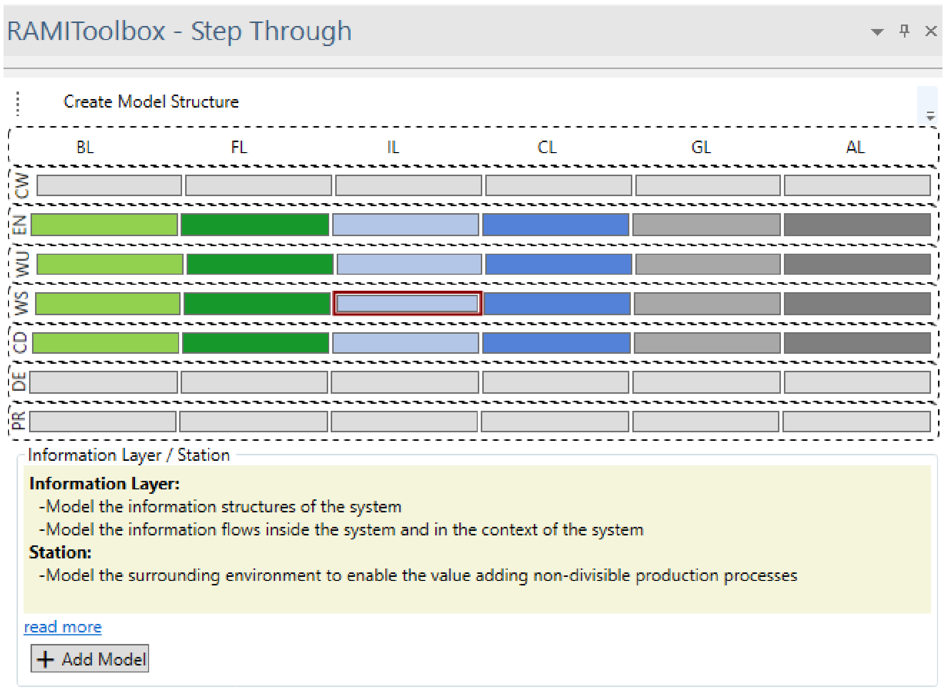
- [ ] Create CW Structure
- [✓] Create EN Structure
- [✓] Create WU Structure
- [✓] Create WS Structure
- [✓] Create CD Structure
- [ ] Create DE Structure
- [ ] Create PR Structure
- [ ] Create * Structure 8

Figure 6.7: Function for creating the model structure

This structure is needed, as each diagram describes a particular part of the system and should be assigned to the corresponding folder. Moreover, multiple event handlers and automation functions rely on this structure to be existing. To not need to create each folder manually, this function sets up all needed packages to describe the system.

In this example, the folder structure for the Enterprise, the Work Unit, the Work Station, and the Control Device level are chosen. However, this should be selected according to the aspects of the system to be described. Usually, selecting only single-level groups is a better option than creating all levels at once, as this contributes to better management of the MBSE process. After selecting the levels to be created and click on OK, the project browser of EA is adjusted accordingly. All folders have been created as Packages within EA. Beyond each package, RAMI 4.0 views have been created from Business Layer to Asset Layer.

A significant advantage of the RAMI Toolbox is that the engineering of flexible production systems can be done bi-directionally. Either the created package within the EA browser could be utilized, and new diagrams could be created, or the toolbox might be applied. If changes happen to one of the respective perspectives, the other will notify and automatically adapt. Figure 6.8 shows one of those scenarios.

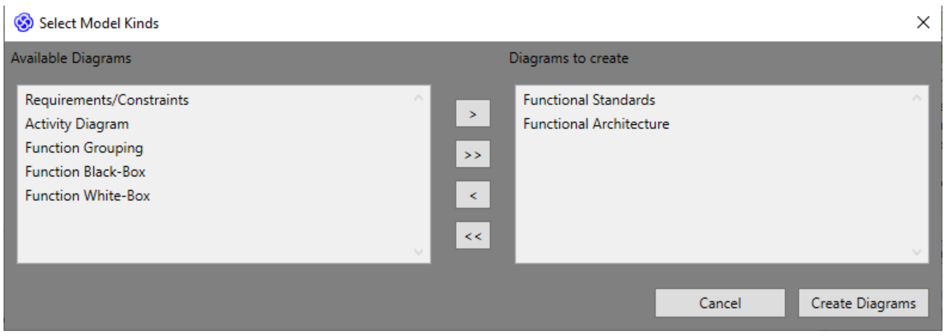


Information Layer:
-Model the information structures of the system
-Model the information flows inside the system and in the context of the system
Station:
-Model the surrounding environment to enable the value adding non-divisible production processes

Figure 6.8: Description of each pane

In this image, the main GUI of the RAMI Toolbox is depicted again. However, it can be seen that the panes currently under consideration changed their colors according to the RAMI 4.0 layers. This helps identify the needed steps for developing the SoI and which views might not be considered. A more detailed description appears by clicking on one of the panes. This description is individual for each pane and gives guidelines for the needed modeling steps within the pain. For example, the Information Layer at the Station level is selected, as seen in the image. Thereby, the window shows what models need to be created for describing the Information Layer, as the information structures or information flows, and what models need to be created to define the Station level, like modeling the environment for enabling the value-adding non-divisible production processes.

At the bottom of the image, a button called “Add Model” realizes the function of automatic model generation. According to the selected pane, this function offers model kinds needed to implement the RAMI 4.0 view at the particular granularity level. While not all model kinds are required to describe the pane, this function restricts the systems engineer only to use suitable models. However, all offered diagrams should be utilized for a comprehensive architecture description. For example, Figure 6.9 depicts the available diagrams for the Function Layer. These include a requirements/constraints diagram, an activity diagram, a functional architecture diagram, functional standards or groupings, and black-box and white-box diagrams. As also seen in the image, any diagram could be chosen to be created, or all of them could be created at once. After clicking on the “Create Diagrams” button, the corresponding diagrams are located within the project browser, where they can be selected for different engineering tasks.



Requirements/Constraints
Activity Diagram
Function Grouping
Function Black-Box
Function White-Box

Functional Standards
Functional Architecture

Figure 6.9: Window for selecting the model kinds

Finally, the RAMI Toolbox main GUI is not the only entry point for selecting functionalities provided by this framework. In addition, the RAMI Toolbox can also be called up via the Specialize-Tab in EA. Single functions are listed that become visible when selecting the RAMI Toolbox option. As currently implemented in version 1.0, this framework offers additional useful functions supporting MBSE of flexible production systems. For example, AutomationML-files could be exported, imported, or synchronized with existing RAMI 4.0 system models. Moreover, XML-files containing commands for PLCs might be exported after describing the Information as well as the Communication Layer of RAMI 4.0. The “PIM $\rightarrow$ PSM” option refers to automatic model transformations. This helps transform the PIM automatically to the PSM without much manual effort. In the end, the bottom three functions provide administrative functionalities, like showing or changing the toolbox window and exhibiting toolbox information or support. These are not applied to any activities in any MBSE activities.

As previously mentioned, the RAMI Toolbox supports automating tedious and repetitive tasks. One example of such a task is the model transformation between the particular RAMI 4.0 layers. For example, the functional requirements need to be transformed into system functions, while logical or physical components realize those functions. Transforming the respective system elements to each other layer is such a tedious task, repeated simultaneously for each modeling element. To support this scenario, the RAMI Toolbox offers a separate window, which can be started via its user interface. As indicated in Figure 6.10, this window consists of two main parts, shown as sub-spaces on the left and right-hand sides.

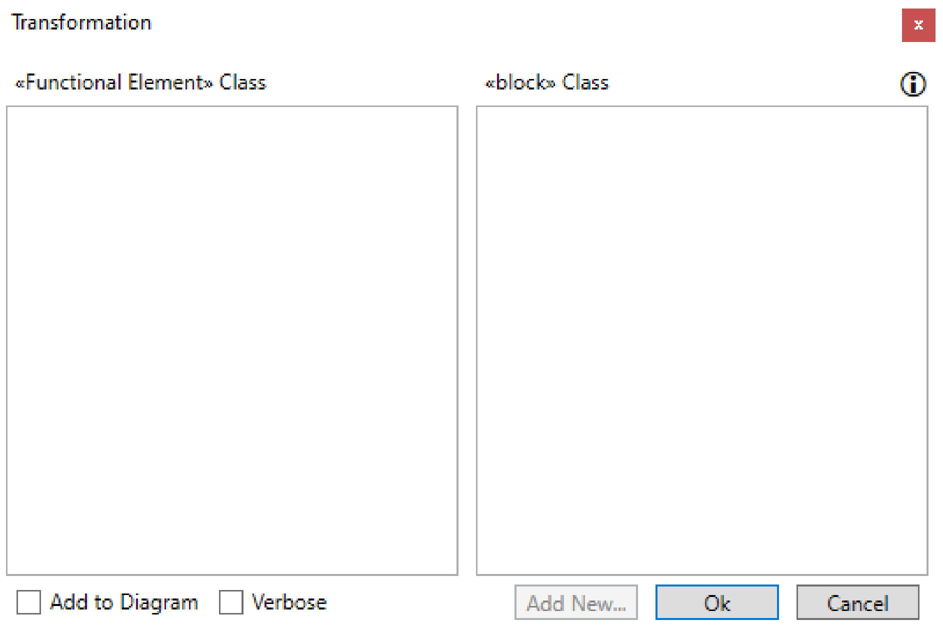


Figure 6.10: Model transformation window

On top of each space, the UML type and the stereotype of the chosen attributes are shown. In this specific example, UML classes from the stereotype block are selected. Those parameters originate from a separate XML file, which stores all configuration information to execute its functionality correctly. If the window starts with the chosen parameters, all elements from the upper layer are shown on the left-hand side, while the right side lists all elements of the lower layer. However, no elements are listed in this scenario to demonstrate the layout of the user interface without giving a particular example. Next, the functionality of this user interface allows interconnecting of the respective elements. This is done by clicking on one or more elements on each side and establishing a connection. Moreover, it is additionally possible to add new system elements to the model within the window. Thus, especially for non-experts, this user interface establishes traceability and performs model-driven engineering with the RAMI 4.0 modeling framework, as all elements are added to the model by themselves. No diagram needs to be adapted manually.

A unique feature of this window is that it might be applied for model-driven engineering and composing or decomposing. While the previously mentioned example explains the possibility of its utilization for transforming requirements to functions or functions to system components, the same stereotyped components could also be decomposed on the equal RAMI 4.0 layer. For example, higher-level requirements could be refined with more specific or derived requirements at lower levels. Regarding functions, white-box functions with all their parts could be decomposed on a lower granularity level by defining all sub-functions that realize those parts. Thereby, the respective black-box functions could be aggregated, and more specific functional architecture could be created. The same counts for logical or physical components. Those system elements build the system on different abstractions. While a factory consists of multiple production lines comprising several machines, such a decomposition is often obligatory when engineering such systems. In conclusion, it could thus be claimed that this user interface is highly supportive and offers primary functionality to ease the task of modeling system architectures of flexible production systems.

Chapter 7

Application of the Framework within the Toolchain

This chapter delineates the RAMI Toolbox's actual application after specifying domain-specific concepts and their implementation. Thereby, different quality attributes could be analyzed based on different application scenarios. Specifically, those scenarios consider co-simulation, model transformations via interfaces, or possibilities of describing external systems.

7.1 Mosaik Co-Simulation

To investigate varying industrial CPS and their interplay during run-time to detect undesired emergent behaviors, an interface between RAMI 4.0 and Mosaik could allow the analysis of modeled systems before their implementation. As Mosaik mainly targets the Smart Grid area, as outlined in Section 2, its utilization for co-simulating modular production islands needs to be examined. However, a successful implementation of the Mosaik interface for flexible production systems could ensure flexibility and adaptability under volatile conditions and reveal the optimization potential of engineered systems before passing them to other tools within the toolchain. Hence, as using a Smart Grid framework in the industrial area appears meaningless at first glance, ensuring this integration tough contributes to multiple aspects (Dragicevic et al., 2020). In more detail, requirements for enabling this co-simulation are the possibility to investigate complex SoS and its parts, which is barely addressed in current production system simulations.

Moreover, created models of such systems might be evaluated before the system implementation, and simulation results could fall back to the model. This means faults and errors are recognized at early engineering stages, an essential requirement for most manufacturers (Carvajal Soto et al., 2019). RTE is a proper function to ensure this bi-directional optimization. However, as the investigation of emergent behavior is barely addressed in recent simulations (Z. Li et al., 2006), simulating such SoS by using co-simulations could exhibit bottlenecks or undesired functionalities. Finally, the interconnection between Mosaik and RAMI 4.0 provides an additional example for setting up an entire engineering toolchain with the RAMI Toolbox, which is the primary objective of this chapter.

The goal of creating an interface between RAMI 4.0 and Mosaik is addressed by utilizing the respective programming interfaces to transmit the information from the architecture to the simulation. Hence, suitable technologies and frameworks must be used to develop software appealing to these specifications. Since the RAMI Toolbox is free, offers various functionalities, and is easy to adapt, it is the tool of choice for implementing an additional method realizing the interface between RAMI 4.0 and Mosaik. To specify this in more detail, the toolbox introduces features concerning usability and automation of recurrent modeling processes. Therefore, the first step is to extend the RAMI Toolbox DSL to consider the information needed for simulating the interconnection of elements within an architectural model. For example, a new element called *MosaikSimulatorConfig* is added. As the name assumes, this element provides configuration data for adjusting the exported simulators in Mosaik. Doing so is derived from the UML *metaclass* Artifact. It extends it with additional attributes for placing the configuration values, which are explained in more detail in the following:

- *fileName:* This attribute is provided by the metaclass Artifact and is used to save the path to the executable simulator file. According to the implementation of the simulator, this file is either a Java or a Python type.
- *confKey:* In this attribute, the connection type for the simulator is described. The simulator could be connected in-process, started, and executed in its thread or linked to another running simulator.
- *confValue:* The information needed for starting the simulator is stated here. The connection type could be a Python class, a terminal command, the IP address, and the port to a running simulator.
- *isModeled:* This Boolean value is set if the simulator is generated out of the model. If true, the functional description in the Primary Use Cases generates code for a simulator. If isModeled is set to false, the simulator itself already exists

and only has to be connected to the Co-Simulation.

- *models:* Additionally, if the attribute isModeled is true, all IDs of assigned Primary Use Cases and additional model information are taken for use, which is stored in this attribute.
- *stepSize:* The number of steps that elapse till the simulator is executed again is represented in this attribute. When generating code from the model, currently, only a fixed step size can be stored, which cannot be changed during runtime.

Simulator Configuration

To secure procedural correctness of the simulation during run-time, the so-called *scenario.py* file of Mosaik characterizes the primary process. The Mosaik framework is initialized by importing the needed modules and creating entities for each simulator class. The configuration is loaded by a file called *config.json*, where basic settings can be placed. The framework's main component is represented by its core, which manages the respective simulators and ensures scheduled processing. Hence, the simulators for the different production islands are instantiated according to the information in the configuration file. This information is previously extracted from the SysML model and information about the interfaces and data exchange between each production island. Those simulators are then added to the simulation scenario during run-time, allowing various control strategies to be simulated. (Schütte et al., 2011)

Thus, a specific user interface is designed to ensure the easy usage of the Co-Simulation integration within an RAMI 4.0-based architectural model. The main intention of this interface is to enable the possibility to configure the previously mentioned attributes and provide additional model information containing Primary Use Cases and links to their respective behaviors. Therefore, the user interface is divided into two parts. On top, general Co-Simulation settings can be found, and on the bottom, the particular settings are stated for those simulators, which are generated based on the architectural model. Additionally, the interface ensures that all inputs are correct, applying them for usage in the subsequent code generation.

Next, the simulators of the modeled system components are developed. Therefore, each simulator consists of three Python files, an application programming interface (API), the simulator itself, and the associated models, explained in detail in the following. While the API deals with implementing the interface of Mosaik, the primary purpose of it is to communicate with the co-simulation master algorithm by providing functionalities and data with the help of a Java wrapper. Hence, the primary purpose of

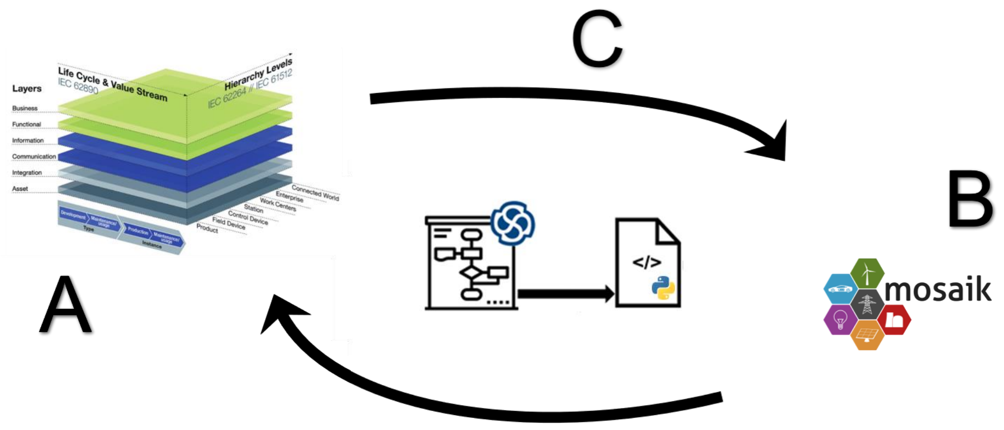


Figure 7.1: Bi-directional Interface between RAMI 4.0 and Mosaik

this wrapper is to provide the Python code to the API of Mosaik. The simulation class itself instantiates all entities of the respective model and administrates its intended functionality. In the models, the business logic and the functionality are constituted, which the simulator will execute during run-time. However, this is supervised by the configuration of the co-simulation, where the start and end conditions are enabled, as well as the treated procedure during each step. (Schütte et al., 2012)

In more detail, the settings on top of the user interface are general settings for the simulator, like the start type, the start command, the step size, the link to the simulator file, and a check box to generate the simulator based on the model. Those values can be stored for later usage for each simulator separately. However, if the simulators need to be generated from the RAMI 4.0 model, the bottom area has to be considered, as illustrated in Figure 7.1. In the left window, all Primary Use Cases related to the current instance of the simulator configuration are listed. Shifting them to the window on the right-hand side, their behavior is exported to the Co-Simulation. Additionally, by double-clicking on each use case, this behavior can be viewed in more detail. Usually, an activity diagram is used to model the sequence of events precisely. Beyond these windows, the detailed configuration of a single simulator model is given by introducing five settings. The model name titles the model as represented in the Co-Simulation, whereas the init values contain the initialization parameters passed to the simulation model at the instantiation. On the other hand, the input and output values define the variables that can be given to a simulator model during its runtime. The last setting specifies the root activity by selecting a primary activity in the drop-down menu.

Code Generation

After setting up the configuration details for all simulators used in the simulation scenario, they must be exported from the model and imported into Mosaik. This is done by developing an additional method to the RAMI 4.0 Toolbox, which is based on previous research results from the Smart Grid domain (Binder et al., 2019). Hence, all considered simulators are first collected from the model by clicking on the Co-Simulation function. After choosing a suitable configuration duration in single steps, the behavioral UML diagrams are a base for future code generation. Technically speaking, in Mosaik, each simulator consists of three Java classes. Consequently, each modeled activity diagram must be transformed into those three classes. Since exporting class diagrams as XML-files is a difficult task, templates of Java classes are created in the first place. The export scenario makes use of these templates and replaces or adds each used code snippet during its application. An easy way of dealing with this is provided by the framework StringTemplate, which enables to set markers in template files and replace those with generated code. Utilizing this method, the init, input, and output values originating from each simulator configuration and the behavioral code described in UML are applied to generate the needed Java classes. However, the generation of functional code based on behavioral diagrams in EA has some restrictions that must be considered. First, all behavioral diagrams must be a child element of the respective class. Secondly, additional Activities are not allowed to be a child element of an Activity. All activities considered for the code generation must be a direct child element of the respective class. To work around these restrictions, the Add-In deals with moving and copying elements within the model so that the structure is in the correct order for each simulator to be exported correctly. Furthermore, the behavioral diagrams must be represented as classes since code generation is restricted to this type of UML diagram. Nevertheless, after exporting the Java classes, an executable .jar file is generated and placed in the Mosaik folder. At this place, Mosaik can access the Java code and utilize it during its Co-Simulation run.

Co-Simulation Execution

Executing the previously created co-simulation scenario in Mosaik is intended to validate and demonstrate the feasibility of the toolchain in the context of Industry 4.0. Thus, the results of one exemplary simulation run are shown in Figure 7.2, where the whole production progress is shown by indicating products in orange and inquired ones in blue. Hence, it is shown that the products in construction strongly increased at the

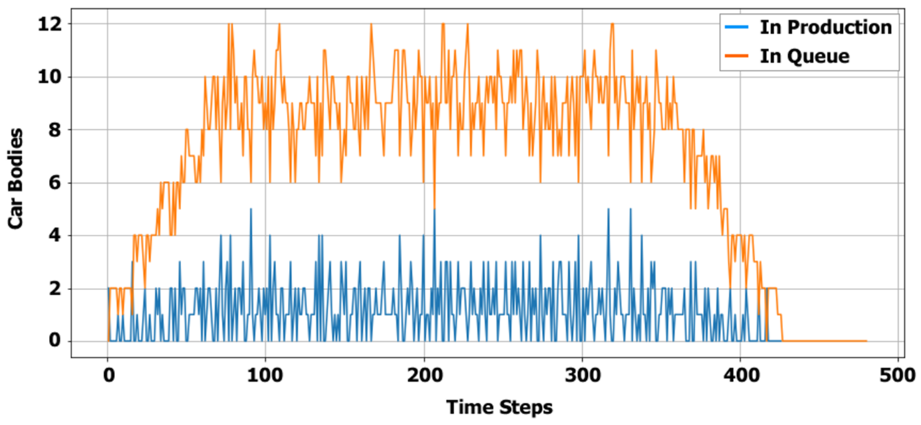


Figure 7.2: Exemplary Simulation Run

start of the simulation, which is traced back to the dependencies between the production islands. Randomly generating specifications for each product will thereby result in different construction times. In addition, two new products are created each 15 time steps, which is initiated by the start of the simulation at time step 0 and ends after ten cars have been instantiated. At around 80 time steps, this behavior evens out because of the exhausted capacities until it finally decreases caused by the production stop after ten instantiated products.

The proposed work demonstrates the general feasibility of using Mosaik in an Industry 4.0 environment and enables the simulation of industrial agents. Each agent may contain an independent and unique behavior, as realized in the model aligned to RAMI 4.0. This allows to observe their interplay in a large-scale area or evaluate their respective functionality during run-time, which is a big step towards handling the complexity while engineering current or future industrial systems. Compared to other state-of-the-art approaches, a unique feature of this work is the flexibility of the contributed approach. To investigate the characteristics of a large-scale and multi-agent industrial system, like recognizing emergent behavior, performing a model evaluation, or executing unit tests, dynamically configuring single elements could be beneficial.

Even though the application substantiates the feasibility of the Industry 4.0 toolchain, the chosen scenario exhibits several limitations. Thus, although reproducible by applying the uploaded material, the work should not be considered a ready-to-use methodology. The contrived approach instead validates the applicability of Mosaik in an industrial environment and does not provide any interpretation of simulation results, which could be elaborated in follow-up projects. Thus, even though the first indications of emergent behavior can be observed in the simulated scenario, the number of vehicles in the applied case study has been too small to make a meaningful statement. In the future, a more sophisticated case study must be used to understand the limitations better quantitatively.

However, as far as the research question of this particular thesis is concerned, it has been shown that comprehensive Co-Simulations could be executed based on previously modeled systems. This either allows to evaluate systems before their implementation within the entire engineering toolchain or interconnect stakeholders from both application domains, system architects, and system tester. This is done by exchanging engineering artifacts between the RAMI Toolbox and Mosaik to ensure traceability between the interdisciplinary tools and profit from each other's results. If something is changed within the model, an adjusted simulation needs to be executed, while simulation results could lead to changes within the RAMI 4.0 model. Thus, although the simple example lacks information to achieve meaningful results, it represents how to ensure holistic and interdisciplinary systems engineering with the RAMI Toolbox based on the domain-specific RAMI 4.0 modeling approach.

7.2 Process Control via OPC UA

In this section, the transformation of a system model toward its actual implementation is further addressed. This allows controlling system components via the modeled architecture by exporting the needed information. In other words, an entire configuration of a HMI could be created based on the previously modeled system architecture. Nevertheless, as this would exceed the scope of this work, a small example indicating how such a configuration could be exported is described in the following. To do so, the communication infrastructure of a complex industrial system is modeled and subsequently transformed into real system components. One of the most promising technologies within this area, OPC UA, is used to execute this model transformation from the PSM towards the PSI. OPC UA is considered the future standard concerning industrial communication and is, therefore, a suitable method for application in this context (Gutierrez-Guerrero and Holgado-Terriza, 2019). While the most promising technology enabling this transformation would be message queuing elemetry transport (MQTT), this standard is used to allow for the interconnection via publish/subscribe messaging (Hunkeler et al., 2008). Automating the model transformation from the technology-specific model of RAMI 4.0 and the implemented system further increases the acceptance of the RAMI Toolbox.

The first step enabling the realization of previously modeled system components is to generate the model itself. Thus, as OPC UA is the technology of choice for modeling and applying the system element, a suitable environment needs to be provided. This is done by adapting the DSL of the RAMI Toolbox so that symbols forming the semantics as well as the syntax OPC UA is available for modeling industrial systems. However, as this standardized communication interface has mainly been introduced for representing exchanged data and used services, the Information and the Communication Layer of RAMI 4.0 appear to be the correct viewpoint for realizing this. Therefore, the following adaptions have been made to enable the modeling of OPC UA with the RAMI Toolbox.

First, the Information Layer uses different data representation methods, data model standards, or technologies. More precisely, this viewpoint represents the connections between the single system elements and the container for exchanging the data. This could be XML, JSON, or any other format. Furthermore, information about the exchanged data is modeled via so-called Tagged Values, which can be considered attributes. Each modeling element contains these attributes and stores information about the sent or received data, while the data itself is modeled in the Tagged Values of the connection. Furthermore, a similar principle is applied in the Communication Layer. Thus, the main difference is that, in this case, the connection type comes into focus. For example, wireless or wired technologies can be used, and protocol like Ethernet or FTP is depicted in the mentioned viewpoint. The needed OPC UA information is again stored in the respective attributes.

A specific gateway system has been implemented to establish the interconnection between the model and the Free Educational Open System Architecture (FREDOSAR) framework. FREDOSAR is considered as middle-ware between model and implementation by providing a reference architecture as well as a reference implementation in the context of IoT (Fischinger et al., 2019). This framework is integrated into the proposed approach, as it ensures RTE between the existing system and its virtual representation. Thus, rather than exporting OPC UA information directly to the HMI of the controller, FREDOSAR is interposed and deals with receiving data from the model and passing it to the implemented system, and vice versa. Thus, rather than implementing bi-directional interfaces, the advantages of FREDOSAR lead to using this framework for consistently exchanging engineering information. To ensure a successful application of the interface, the Communication Layer defines the used communication technologies, depicted in Figure 7.3.

All functions and functional requirements must be assigned to Industry 4.0 conform digital twins of the assets. The digital twins are integral parts of the architecture. This

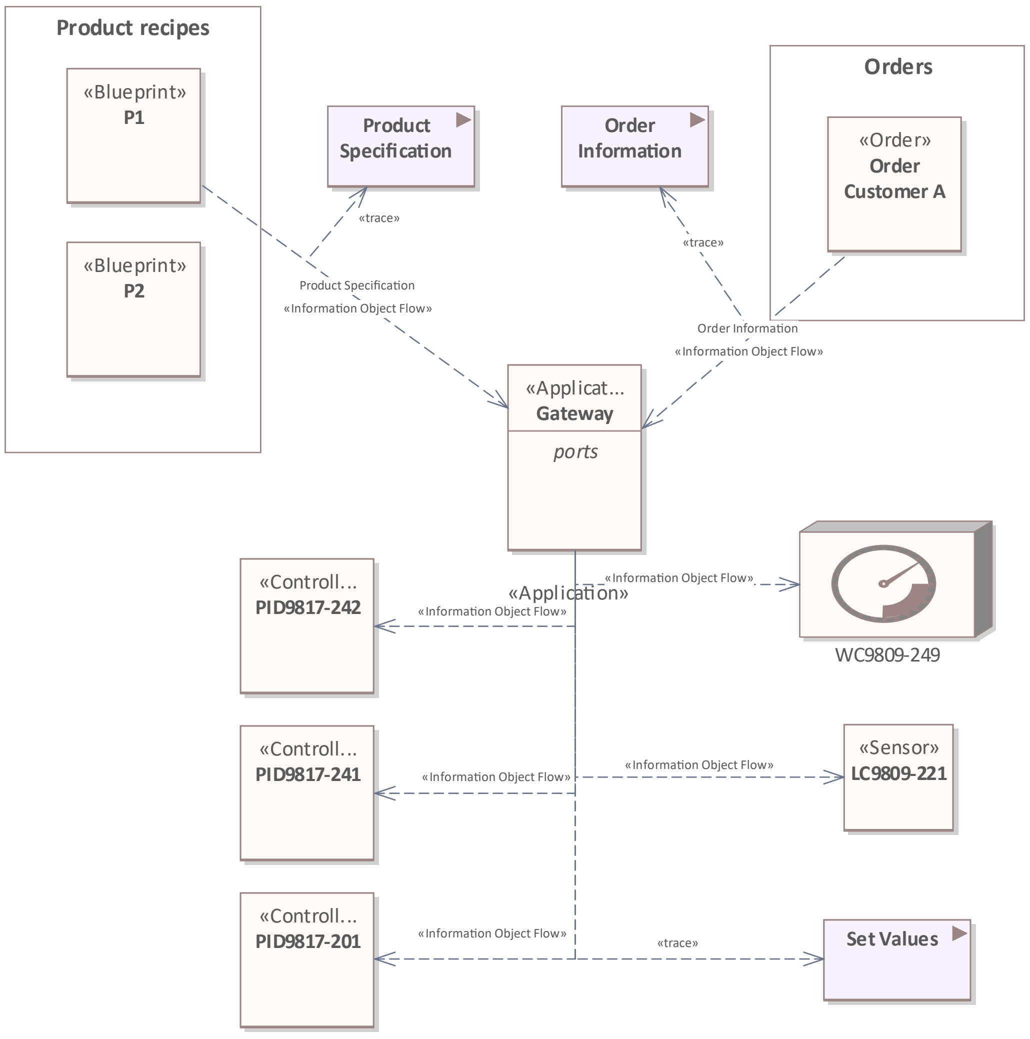


Figure 7.3: Information diagram ensuring OPC UA RTE

assignment is modeled within the Integration Layer. The connections used for modeling the logical architecture indicate dependencies and associations between the elements. As aforementioned, communication and data exchange is defined within the Information and Communication Layer. Finally, all digital objects are assigned to their real counterparts within the Asset Layer. In the next step, the modeled information is exported and transferred into physical elements with the help of the previously developed gateway system. The OPC UA server is implemented in FREDOSAR. In this example, the information exchange is contained within the model in Figure 7.4, especially within the Information and Communication Layer. By adding and registering the exported services, other permitted bundles can access it. Concluded, the newly resulting bundle

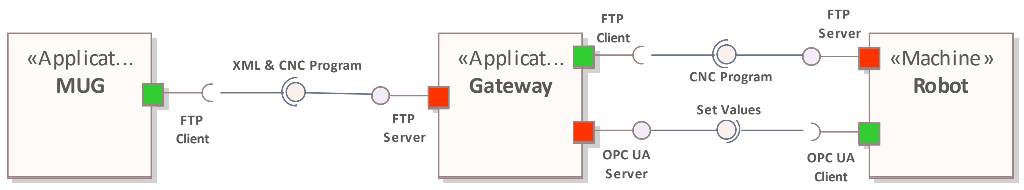


Figure 7.4: Communication diagram ensuring OPC UA RTE

can be integrated and registered with the FREDOSAR environment and, after that, has full access to the OPC UA services that were registered before.

With regard to the research question, exporting OPC UA elements from the model allows controlling functional units or controllers without executing SCADA code or directly accessing PLCs. Thus, stakeholders not involved in this detail level could prepare models describing the intended functionality. This bi-directional interface enabling RTE facilitates comprehensive engineering throughout the entire toolchain. This contributes to low-code or even no-code approaches and traces implementation to requirements at an earlier engineering stage.

7.3 PPR System Description

To also consider the already established definition of PPR systems, the RAMI Toolbox should consider this notation and allow users to model such types of systems. As RAMI 4.0 itself considers flexible production systems, which means smart products, production processes, and manufacturing systems, integrating the concepts of PPR might be a valuable step. This implementation provides users with a modeling method to describe and subsequently process such systems in various environments. By enabling this possibility, it enhances the traditional way of modeling PPR systems (Meixner et al., 2020), mainly done with Microsoft tools. However, those tools exhibit a significant disadvantage, as they only allow modeling on one separate sheet, which is instead considered creating a diagram than a system model. This means if the system becomes more and more complex, including more comprehensive processes or a large number of system components, creating only one diagram of it becomes a difficult task when all aspects need to be visualized appropriately. On the one hand, it is almost impossible to consider specific aspects of the system, while keeping the overview with many connections is barely achievable. Those issues could be counteracted with the PPR integration of the RAMI 4.0 modeling framework. The different viewpoints and granularity levels allow a distinguishing into more granular aspects but ensure the traceability within the model by providing suitable modeling elements.

However, another important aspect is also addressed by this integration. The notation mainly offers elements representing instantiated system components, like process steps or resources. If new system components need to be elaborated, PPR does not provide possibilities to utilize adequate engineering methods. Hence, the RAMI 4.0 modeling framework might fill this gap by previously enabling the development of Greenfield or Brownfield systems before representing them as PPR systems. This means that the RAMI 4.0 concepts need to be traced to the ones of PPR to implement this integration successfully, described in the following paragraphs.

The first thing to achieve this step is the integration of the PPR elements into the RAMI 4.0 DSL. In more detail, modeling elements for describing products, processes, and resources need to exist. However, to consider future utilization, skills are also considered, as those might play a significant role in defining prospective PPR systems. In addition to the modeling elements, suitable relationships must be provided. Those relationships build the interconnection between the products and processes as well as processes and resources. Thus, Table 7.1 outlines the implemented PPR elements and relationships of the RAMI 4.0 modeling framework and their underlying UML element.

Table 7.1: DSL including PPR elements and relationships

Type	PPR	UML
Element	Product Asset	Class
Element	Process	Class
Element	Skill	Class
Element	Resource	Class
Relationship	PPR Flow	Control Flow
Relationship	PPR Connection	Connector
Relationship	Resource Decomposition	Aggregation
Relationship	Process Decomposition	Aggregation

The table shows that the four mentioned concepts had been implemented as UML classes, while four different types of relationships are available. More detailed, the PPR Flow deals with delineating the process flow of the system and how the processes modify the products. Thus, it is derived from the UML control flow. Additionally, the PPR connection connects the process with resources. It indicates which resource is used to fulfill which process by simply utilizing a connector. Finally, resource and process decomposition traces the mentioned components on multiple granularity levels of RAMI 4.0. Those relationships are solely used to ensure traceability and consider the “abstraction of concerns” and “divide and conquer” principles. Therefore, as the resources or products from the lower granularity are parts of those from the higher granularity, an UML aggregation is used to indicate those hierarchies.

Implementing the respective PPR elements as UML class combines the advantages of both representations. While classes are general elements within the UML modeling language, many extension possibilities are available. Thereby, class attributes could be added to point out the individual character of each of the PPR elements might be added, which additional tagged values could also support. For example, a product asset could be enhanced with values describing the product, like measurements, production characteristics, or compartments. As far as a process is concerned, those attributes could represent input or output for each process step. Thereby, comprehensive PPR process chains could be modeled. The most crucial interconnection between UML and PPR is fulfilled by implementing resources as those elements build the factory hierarchy trees and thus represent the actual production networks of the manufacturing companies. Furthermore, resources deal with exporting engineering information to other tools within the toolchain, outlined in detail in the next chapter. However, suitable attributes need to be elaborated for those elements as well. For example, a resource could additionally contain attributes for calculating KPIs or the types of machines utilized. Therefore, suitable values need to be examined, which are deposited within the DSL elements of the RAMI 4.0 DSL and made applicable for further utilization.

Figure 7.5 indicates the shapes of the respective PPR modeling elements. Similar layouts and colors are used to maintain consistency with the original PPR shapes. However, EA is limited in creating shapes for modeling elements; approximations to reach the original layout have been made. The depicted result thus gives an idea of the implemented PPR modeling element shapes. In detail, a Product Asset is circular and light blue, just as it initially formed. Nevertheless, the process is rectangular shaped and exhibits a transition from white to light blue in terms of color. As far as skill and resource are concerned, both elements are rectangular and continuously light blue. To distinguish them from the process element, corners are more rounded, especially when considering the skill element.

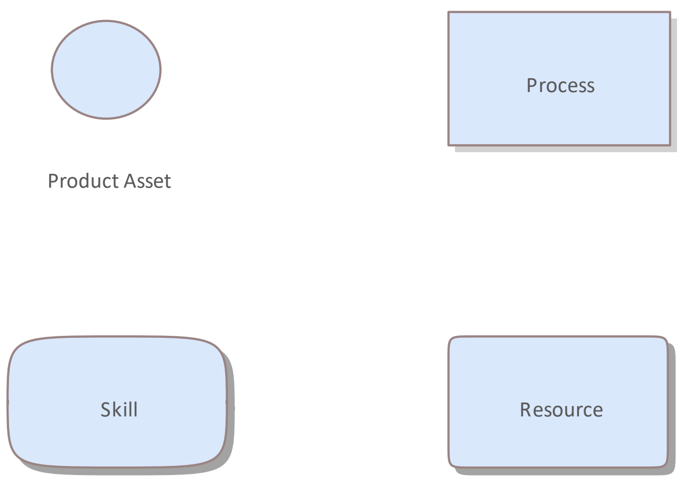


Figure 7.5: PPR Modeling Elements of the DSL

Finally, the respective modeling elements are traced to ensure the interoperability of PPR with the engineering possibilities of the RAMI 4.0 modeling framework. The elaborated functions on the Function Layer could be considered processes executed by the resources. Thus, after defining all functions, processes might be derived. Regarding resources, those are the elements of the instantiated production system. This means these elements could be compared to the SysML blocks of the RAMI 4.0 DSL, as they are used on the integration of the Asset Layer. Thus, the developed system components could be reused as resources within PPR when applying model-driven engineering with the modeling framework. After modeling systems along the RAMI 4.0 layers and hierarchy levels, the respective elements might be directly traced to PPR elements. However, on the other hand, modeling PPR system by applying the integrated concepts could be independently used apart from RAMI 4.0, as the modeling framework enables the possibility to execute this task on its own.

Regarding the research question of this thesis, the PPR integration has shown that it is possible to use different modeling notations to represent the same information. This means various stakeholders could remain in their expertise and use their established methods while the modeling database remains consistent for all practitioners. If something changes within the RAMI 4.0 model, the changes are also traced to the PPR model. On the other hand, while some stakeholders are familiar with the PPR concept, they could model systems according to this notation, which is subsequently traced to the RAMI 4.0 layers. This, for example, allows additional engineering information or new system objects to already existing system infrastructures. Concluding, interconnecting the notations of RAMI 4.0 and PPR allows exploiting each other's advantages, like usage in various application scenarios, by diminishing respective disadvantages, like missing engineering possibilities.

7.4 Information Exchange with AutomationML

This section uses AutomationML for fully automated model transformations from the logical architecture to the technical architecture of industrial systems. Within AutomationML, company-specific SystemUnitClasses are applied, providing metadata for each modeled system element in extensive databases.

7.4.1 Model Transformation by applying AutomationML

The first step towards successfully fulfilling this approach is the need for an AutomationML file. This file contains all available system elements and their stereotypes with information about different attributes. Based on this information, the optimal solutions for the previously elaborated requirements are selected. To do so, the AutomationML file consists of various libraries. As defined by AutomationML consortium (2014), standard role classes are required to model basic AutomationML concepts. Such a class defines the abstract functionality but does not specify a technical implementation. Thus, the role class library depicts all the needed information about the system elements. This also includes the attributes above but no concrete values. Another role class is created for each system element, as different types contain different attributes. However, in the system unit class library, the technical implementations of each abstract role class find their place. This means the concrete system elements are derived from the respective class, and the corresponding values are assigned to the attributes. Those values, therefore, are the basis for future solution selection from the algorithm. According to the optimal match, the individual system element is suggested to the engineer.

The current specification of the RAMI Toolbox already inherits a usable metamodel for developing industrial systems according to RAMI 4.0. However, adjustments to the provided modeling elements must be made to work with the intended approach. More precisely, especially the Function Layer in the RAMI 4.0 metamodel has to be adapted, which has been done as described in the following. Accordingly, each function representing a system element must include attributes about desired dimensions, available voltages, or the required rotation speed. The concrete values can be derived from the requirements or calculated from specific mathematical calculations. The results are then embedded into the respective attributes and added to the metamodel representation of RAMI 4.0.

The last artifact to enable automated model transformations is the algorithm itself. Implemented within the RAMI Toolbox, the functionality must follow a specific process. At first, the required attributes are gathered from the logical architecture of the RAMI 4.0 model. Then, according to these values, the AutomationML file is iterated, and the optimized solution is elaborated. Several trade-offs are considered if no optimal solution exists or more than one exists. Those are presented to the engineer, which ultimately makes a decision. Subsequently, after the right system element has been selected, it is automatedly created within the technical architecture of the model and traced to the respective function it fulfills. This ensures traceability within the industrial system and is an essential step towards dealing with complexity in such a critical infrastructure (Sindico et al., 2012). However, when each technical solution has been selected and again implemented into the industrial model, the PSM of the system can be modeled by utilizing the applied hardware components. Again, this process is described more precisely in the following section.

7.4.2 Round-trip Engineering by applying AutomationML

As more and more complexity emerges during the development of flexible production systems, multiple engineering tools need to work together. Each addresses a separate part of the complex system, which reduces the complexity by breaking the system down. To close this gap in current approaches, a bi-directional interface between RAMI 4.0 and AutomationML could enable the interconnection of models within the toolchain. While developing such a system could be achieved with RAMI 4.0 and the RAMI Toolbox, the resulting system model could be transferred to other engineering tools with AutomationML. Thus, a bi-directional interface automatically exports AutomationML-files from previously modeled systems or imports SysML block definition diagrams from once-created AutomationML-files.

Before developing the interface between the respective modeling frameworks, similar concepts describing the same subject must be compared and mapped if needed. While the upper layers of the RAMI 4.0 modeling framework, which is implemented in the RAMI Toolbox, describe contextual aspects with various domain-specific languages, the bottom layers are implemented with well-known UML diagrams or the SysML. This means the technical system is decomposed into its single part at the end of the engineering process by applying an SysML block definition diagram. In conclusion, this type of diagram appears to be the best match to be transformed into AutomationML and vice versa. Thus, Table 7.2 shows the results of the mapping between SysML and AutomationML. The proposals of Berardinelli et al. (2016) are also considered, as they introduce their stereotype to link the respective concepts instead of mapping them directly.

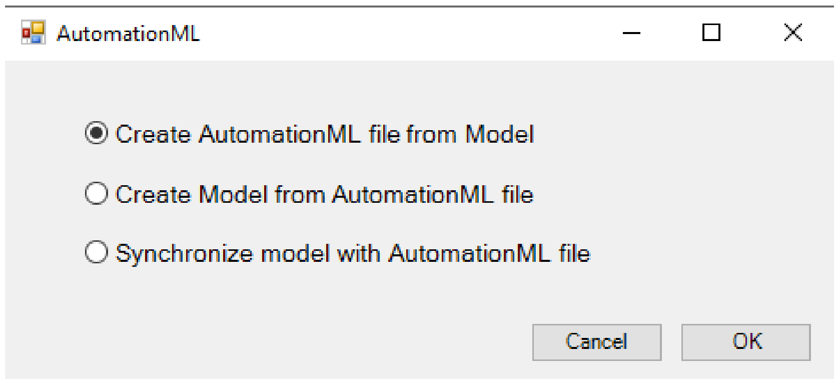
Table 7.2: Mapping between AutomationML and SysML concepts

Concept	AutomationML	SysML
Model	File	Model
Libraries	InstanceHierarchy	Package
Objects	InternalElement	Block
Interfaces	ExternalInterface	Port
Attributes	Attribute	Tag
Abstraction	Decomposition	Part Association
Connectors	InternalLink	Connector
Pattern	Role	Stereotype

The table indicates that EA modeling elements could be equally transferred to AutomationML files, representing the same physical component with another presentation. Following this principle, each InstanceHierarchy of AutomationML is realized with a Package in EA. Those elements represent roots for storing modeling elements or system components. Moreover, SysML Blocks are translated to InternalElements, meaning a factory's actual components. Those components inherit all attributes and are aligned within a hierarchical tree structure. To give further examples, Ports in EA realize the ExternalInterfaces, which deal with the interconnection between the system components. Thereby, ports could also inherit connection information as well as interface realizations. As far as Attributes are concerned, those AutomationML concepts represent the same aspects as Tagged Values in EA. InternalLinks are realized within the model with Connector relationships, and the decomposition between the elements is extracted from the part associations. Finally, element roles are implemented by stereotypes within EA. Those stereotypes deal with providing classifications for system elements. For example, different machines consist of other stereotypes according to contained attributes or fulfilled functionalities.

After mapping the respective concepts, the interface itself could be implemented.

As seen in Figure 7.6, this is done by providing a new function via the RAMI Toolbox GUI. This function realizes the bi-directional interface and offers an import and export functionality. While explaining the interface's complete source code would exceed this section's scope, the main functionality is outlined roughly. The export function is called on a SysML block and recursively finds all connected elements, ports, and attributes according to the previously defined correlations. The enclosing package deals as InstanceHierarchy to store all plant information. After seeing the corresponding elements of the chosen block, a new AutomationML file is created, and the listed elements are inserted one by one.



● Create AutomationML file from Model
○ Create Model from AutomationML file
○ Synchronize model with AutomationML file

Figure 7.6: RAMI Toolbox interface enabling RTE

Therefore, Drath (2012) proposed a C# API that can automate this step with minimal manual effort. This API can directly be implemented into the RAMI Toolbox as dynamic-link library (DLL) since the RAMI Toolbox itself is implemented in C#. Thanks to this DLL, complex XML-transformations are abolished, and resource consumption is optimized. After creating the AutomationML-file, it is saved to the designated storage space, where it could be used for further processing in other tools.

The counterpart, importing an AutomationML-file into an EA model, follows the same principle. After choosing the import function via the RAMI Toolbox GUI, an AutomationML-file could be selected. This file is subsequently traversed, and all InternalElements with their correlations are stored within the EA package. Thereby, a particular focus is also set on stereotypes. If a specific system component has already been derived from such a stereotype, stored in the AutomationML file as a role, those stereotypes are again assigned to those elements. This allows to export and, subsequently, import such information as well. A separate SysML block definition diagram is created, which displays all imported blocks, ports, attributes, and connections. This allows directly using the imported elements to be interconnected with the already existing RAMI 4.0 model or further editing them with the EA modeling tools. If the system's architecture has been described with the modeling framework, the single elements could be processed in external tools of the toolchain. After successfully adjusting those elements to detailed engineering details, which could include the addition of electrical information or the completion of mechanical attributes, those elements and the accompanied data could be re-imported into the original system architecture, where additional process steps originating from basic engineering might be triggered.

With the help of the interface, the resulting SysML block can be exported and externally processed, for example, with the AML Editor, or the resulting AutomationML-file can be imported again into the industrial plant model. After execution, all related blocks are also implemented as InternalElements, which counts for the ports respectively ExternalInterfaces. The hierarchy between the elements has also been correctly implemented. In addition, the attributes are also successfully transferred either by importing or exporting them. However, as currently only instantiated systems can be considered within RAMI 4.0, only the InstanceHierarchy could be investigated. This also entails that the corresponding supported RoleClasses, SystemUnitClasses, or InterfaceClasses could not be linked to the existing instances. Figure 7.6 shows the functionality that the RAMI Toolbox provides to enable RTE.

The interface itself provides three different choices. The two choices on a top deal with exporting models to AutomationML files or the other way round. However, the third functionality, synchronizing the model with the AutomationML file, is the one that is needed to enable RTE of RAMI 4.0-based system models describing flexible production systems. This function allows to synchronization of independently adapted system models and updates them from each other's adjustments. Thus, it provides the foundation for the bi-directional model to model transformation in the context of RTE. In detail, when executing this function, the SysML component tree is recursively iterated through, and all components are listed. Subsequently, an AutomationML file is chosen, where all included system components are also discovered. The next step deals with comparing the respective components and finding the missing ones in each model. This is done for each component part, like interfaces, attributes, or sub-components on lower granularity levels. However, suppose one of the mentioned aspects is missing in the system model or the AutomationML-file. In that case, this functionality creates new model elements and links them to the existing model in each tool. Thus, the respective system models are consistent with the implemented RTE approach.

Regarding the research question, this functionality provides a significant benefit.

As the RAMI Toolbox is mainly applied to perform basic engineering tasks, i.e., to create an abstract system representation, this information needs to be enhanced within various detailed engineering disciplines. Thus, the first part of the question is mainly addressed, which falls back to AutomationML being a suitable standard for consolidating domain-specific peculiarities from various engineering domains. At the same time, RAMI 4.0 enables model-based engineering of industrial systems. Aligning the respective approaches ensures holistic and interdisciplinary engineering throughout the whole system life-cycle. The proposed interface thereby gives consistency. The system's architecture is provided to the various disciplines, while AutomationML deals as a central model between them. If one specific discipline leads to changing the system model, this could be adapted within RAMI 4.0 as well.

Chapter 8

Evaluation

The previously outlined chapters deal with three significant aspects to fulfill the research goal of this thesis. At first, the development of domain-specific concepts to describe PPR systems was outlined, followed by the actual implementation of the modeling framework. At last, the integration of the framework into the engineering toolchain has been delineated in detail. This resulted in analyzing various detailed concepts to provide a comprehensive solution. The outcome provides systems engineers with a modeling environment to help them describe or develop their respective systems. With the integration of an extensive toolset, the RAMI Toolbox might address different scenarios. However, as specified in the research questions, the modeling framework should holistically use the concepts of RAMI 4.0 and MBSE to develop current and future production systems. To which extent the proposed solution can fulfill these requirements is evaluated in the remainder of this section.

By applying the concepts of Hevner and Chatterjee (2010) for performing research in the context of this thesis, the aspects of design science research in information systems are used. Among seven guidelines, researchers are required to evaluate developed design artifacts rigorously by providing five different evaluation methods, which include observational, analytical, experimental, testing, and descriptive evaluation; proper guidelines for choosing the correct evaluation strategy for a given scenario are not offered. However, evaluation in the context of this thesis is a difficult task to be achieved. Usually, to evaluate obtained artifacts in utilization within their final application environment, engineers that apply those artifacts need to be observed, and the results should be compared to traditionally used approaches. This exhibits the additional value of the research outcome to previous applications. Nevertheless, as Industry 4.0 describes a new trend and detaches those traditional applications, a comparison against them would be meaningless. In conclusion, this implies that new evaluation strategies must be applied to validate this thesis's outcome.

Evaluating the developed artifacts is crucial to enable an adequate interpretation of the obtained results concurrently regarding the needed rigor. Thus, evaluation is essential in information systems and design science research (Klecun and Cornford, 2005). Thereby, one of two perspectives is chosen regarding the evaluation strategy. Within the *ex post* perspective, a system or a technology is evaluated after it is acquired or implemented. Thus, this strategy is more common in traditional evaluation scenarios. However, the *ex post* perspective considers systems before their implementation. The evaluation strategy used within this thesis will use both viewpoints. Thereby, the proposed framework of Pries-Heje et al. (2008) is taken into account, suggesting a combination of both perspectives to validate all research artifacts comprehensively.

Peffers et al. (2012) further investigated the topic of design science research evaluation, as the consensus view is that the rigorous evaluation of design science artifacts is essential. As there are various types of artifacts and many forms of evaluation, a concept for guiding which evaluation strategy to which artifact is needed. Thus, in their work, the authors analyzed a variety of publications in different journals and indicated that several popular combinations of artifacts and associated evaluation methods exist. For example, the outcome shows that algorithms should be evaluated with the help of technical experiments, while frameworks should be evaluated using illustrative scenarios. In addition to elaborating on these popular combinations, the authors designed a method to facilitate data collection, which overcomes the limitations of traditional bottom-up planning processes. Based on the analysis results, such methods should be evaluated by relevant case studies. Their work is exemplary and shows how such a method evaluation by utilizing two case studies could look. Hence, the evaluation strategy of the three significant artifacts falls back on the proposed artifact evaluation-strategy combinations of (Peffers et al., 2012).

On the other hand, the practically applied evaluation utilizes the concepts of SAAM, a widespread methodology for analyzing developed software architectures. As defined by this method, varying application scenarios need to be specified and applied to enable prototype implementations and provide proof-of-concepts (PoCs). By doing so, it is validated whether the proposed modeling framework can be used for its determined usage purposes. Since this thesis aims to model current and future production systems architectures, typical application scenarios have been defined. In detail, those are:

- Utilization of the RAMI 4.0 modeling framework to describe Industry 4.0-based systems with all domain-specific aspects and follow the development process guidelines. Peffers et al. (2012) suggest that the framework should be evaluated with the help of an illustrative scenario. Thus, the evaluation is based on the metal profiles for the subway track case study, which is more precisely described in Section 8.1.
- Application of the hierarchical matrix structure of the modeling framework to develop the technical industrial plant system. Thereby, different scenarios are investigated, like model-driven engineering of Greenfields or Brownfields and digital twin development of already implemented systems. According to Peffers et al. (2012), this method could be evaluated by applying a case study, which is represented by the Fischertechnik Smart factory case study and delineated in Section 8.2.
- The integration of instantiated models, based on the RAMI 4.0 modeling framework, into an engineering toolchain is validated by analyzing the possibility of exchanging architecture information with other tools or RTE of developed system architectures. This is done using AutomationML and enabling the modeling of PPR systems, thereby representing the model's engineering artifact. According to Peffers et al. (2012), such models might usually be evaluated with technical experiments and illustrative scenarios. Thus, the obtained artifacts are eventually evaluated by applying the exemplary scenario of a packaging station and possible experimental implementations of technical interfaces, outlined in detail in Section 8.3.

The application of the respective application scenarios and concurrent evaluation of the RAMI 4.0 modeling framework is outlined in the remainder of this chapter. Subsequently, the identified findings are stated in a separate section.

8.1 Application of the domain-specific concepts

In the following, the metal profiles for subway tracks case study involving the development process for industrial applications will be outlined. By doing so, the application context for the MDA-approach described in the previous section and design decisions resulting from each phase of the development process are explained. Finally, the concepts and experiences from this case study will be discussed regarding their applicability to evolve a standard development approach for Industry 4.0. According to the considerations mentioned before, the modeling of the case study example follows the

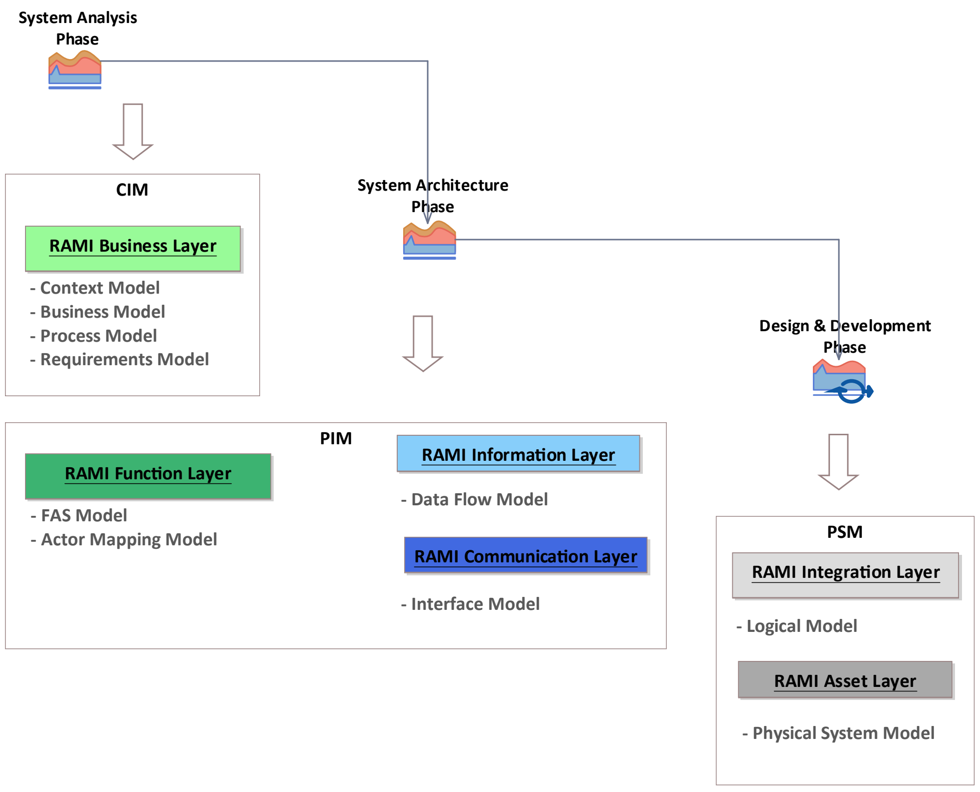


Figure 8.1: Applied process and models of the subway track use case

steps of the MDA-based development process. However, for modeling this system in the context of RAMI 4.0, a specifically tailored DSL is applied. The order of applied engineering steps and used models is outlined in Figure 8.1.

The Business Analysis Process realizes the first step of the process model. This is the task of defining business actors with their respective goals and the enclosing system boundaries, as visualized in Figure 8.2. In this example, the Business Layer inherits five significant actors: the customer, the manufacturer, the raw material supplier, the marketing company, and the machine provider. Each of these actors is connected to one or more Business Use Case (BUC), representing the interconnection and executed business processes between them. However, since RAMI 4.0 defines several value-creation chains, the BUCs can be aligned according to the objectives they try to achieve. For example, the BUC “Produce Subway Track” is part of the value-creation chain “Production,” whereas “Design Subway Track” belongs to “Product Development.” Subsequently, the system requirements are elaborated. This is done during the Stakeholder Needs Definition Process. Therefore, a single BUC is decomposed into several

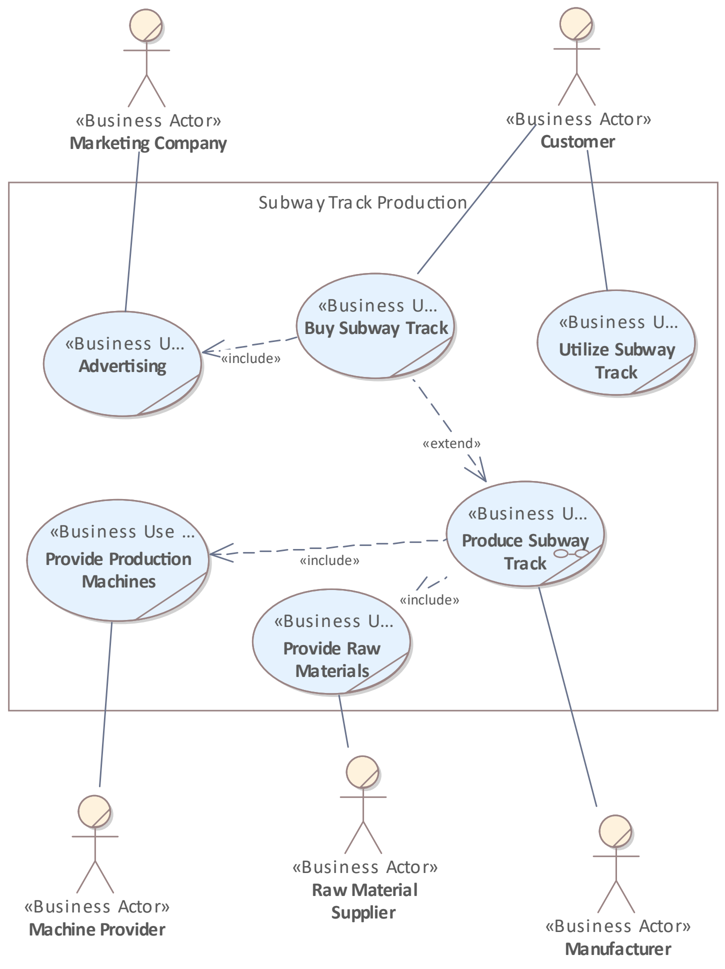


Figure 8.2: Business case diagram of the subway track use case

High-Level Use Cases (HLUCs), specifying the realization of functions performed by the BUC. During this process, it is essential to consider the business goals of each actor, which have been previously defined. One HLUCs therefore builds the base for one or more requirements the system must fulfill. Those requirements are determined by evaluating the intended functionality of these use cases or originate from a requirement engineer talking to architecture stakeholders. However, these two phases combined deliver the CIM as introduced by MDA.

In the next phase, the Requirements Analysis Process, the previously specified requirements are used to elaborate the system's single components. This process is derived from the FAS Methodology. Summarized, one use case realizes each requirement by describing it with an activity diagram. Consecutively, one or more activities can be grouped regarding their intended purpose. This means each functional group represents one component of the system. However, those components are represented by Logical Actors in depicting the Logical Architecture. The interconnection between the actors is realized by use cases, describing the system's functionality. Concerning MDA, the relation between the Business and Function Layer of RAMI 4.0 is realized by a model transformation, tracing Business Actors into Logical Actors and describing requirements with use cases.

After the functional architecture is specified, the Architectural Design Process is executed. More precisely, the Information and Communication Layers describe which data the functions need and how it is exchanged. Therefore, firstly it must be specified which Logical Actor contains what data. Furthermore, the input and output of the executed functions must be defined to process the data correctly. A typical example of Industry 4.0 would be the automatic processing of a product order. After receiving the order, its data must be available for the involved machines to decide whether it is possible to manufacture the desired product with the available resources or if the order has to be delayed or declined. Therefore, the required data must be provided by a system component accessible by other manufacturing components. To process this data correctly, those system components must be connected, specified in the Communication Layer. Therefore, the type of connections between the single components is determined by modeling the interfaces, which are realized as ports. Since RAMI 4.0 is described as a SoA, every component that needs data includes a request point, whereas every component that provides data includes a service point. Summarized, those three layers resulting from the two intermediate phases of the development process build the PIM of MDA.

After the architectural composition of the system is done, the Design Definition Process delivers the Integration Layer, and its MDA's adequate the PSM. As the name assumes, this viewpoint provides a detailed view of the system. By doing so, the system components are modeled based on real-world elements containing as many details as necessary and including all subcomponents to provide information on each abstraction level. For example, the subway track includes sensors, wiring, a control unit, and a condition monitoring unit. Those units consist of various modules or connecting elements, as seen in Figure 8.3. However, the respective subcomponents are defined by applying the FAS methodology to specified higher-level system components. In this way, the different abstraction levels can be considered by modeling the system. Finally, the Implementation deals with the actual application of the system and its components. Suitable approaches for this case would be the utilization of AutomationML and OPC UA.

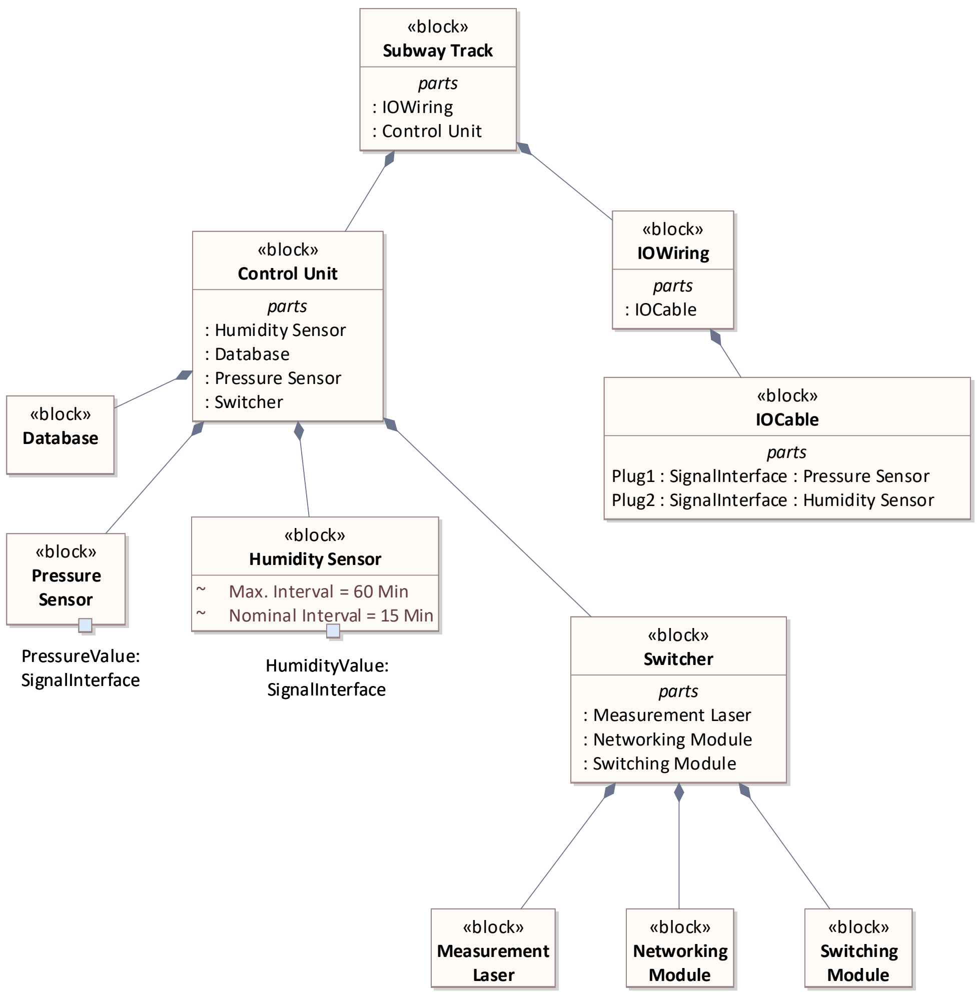


Figure 8.3: SysML block definition diagram of the subway track use case

8.2 Modeling Framework Evaluation

This section outlines the modeled case study of the Siemens Fischertechnik plant. The primary purpose of this model is to evaluate the usability and applicability of the architecture modeling framework. A secondary goal is to provide an example of how to actually apply MBSE to transform an industrial plant towards the needs of Industry 4.0. Thus, the following the transformation of the Fischertechnik manufacturing plant from its original state towards becoming a flexible production system is described in detail. This is done by defining the architecture of the Fischertechnik system with different models, which are divided into an AS-IS (Brownfield) and a TO-BE (Greenfield)

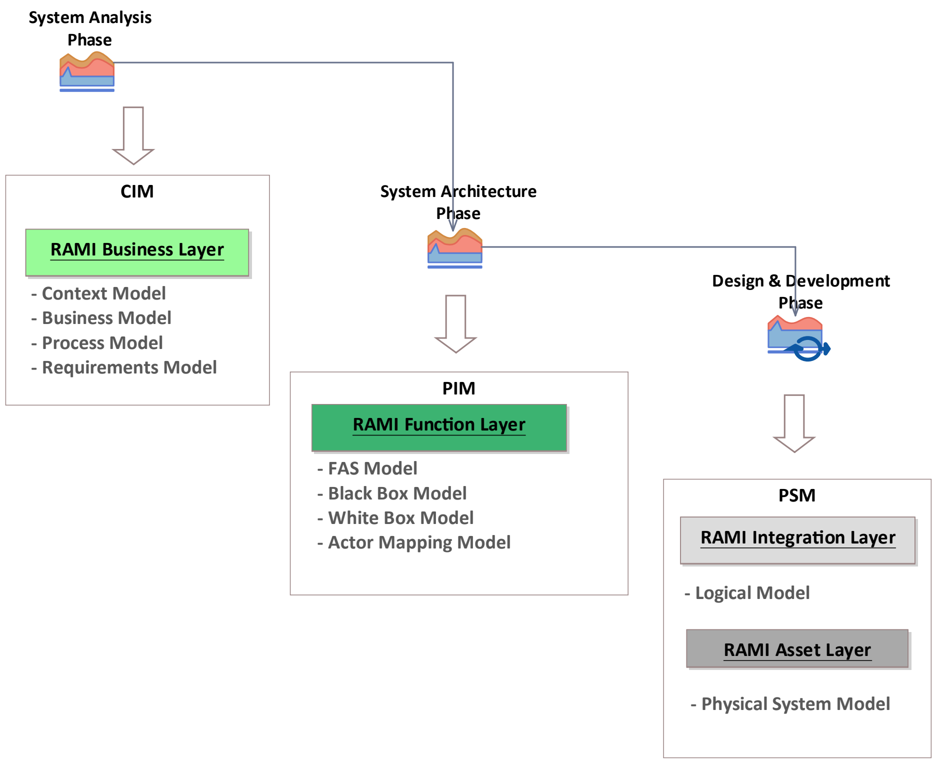


Figure 8.4: Applied process and models of the plastic housing use case

architecture. The model-driven engineering is applied based on the currently used production system in the AS-IS architecture. This system needs to be transformed into a flexible production system allowing manufacturing in lot size 1. The novel Fischertechnik plant should also produce plastic housings instead of cylinder heads. The transformation of this system by utilizing the RAMI 4.0 modeling frameworks is described in the context of Brownfield engineering. However, as far as Greenfield Engineering with the modeling framework is concerned, a novel punching station needs to be integrated into the Fischertechnik system. This is part of the TO-BE architecture and will be addressed subsequently.

8.2.1 Model-driven Engineering of a Brownfield

The first step in describing the current Fischertechnik system is modeling the AS-IS architecture with the proposed modeling framework. The order of applied engineering steps and used models for this use case is outlined in Figure 8.4, which are described in detail in the following.

As mentioned before, the AS-IS architecture is primarily modeled to transform the currently used production system consecutively and ensure more flexible production of plastic housings. As the plant initially had been set up to manufacture cylinder heads, most of this existing system could be reused. Therefore, the following steps for modeling the AS-IS architecture are explained in detail. This is done by utilizing the RAMI 4.0 modeling framework described in the remainder of this subsection. Thereby, a step-by-step guide is delineated, starting with the context model.

Business Layer

In the context diagram, the system context of the SoI is modeled by illustrating the surrounding environment of the system and the in- as well as outputs with other systems, like the delivered raw material or the finished cylinder heads. Therefore, a system context DSL is implemented and used within the RAMI Toolbox. The DSL implements the peculiarities of the SIPOC method, which stands for suppliers, input, process, output, and customer. In the center, the process is considered a black-box and represents all inputs and outputs of this process. How the input is transformed into output is elaborated afterward with different models.

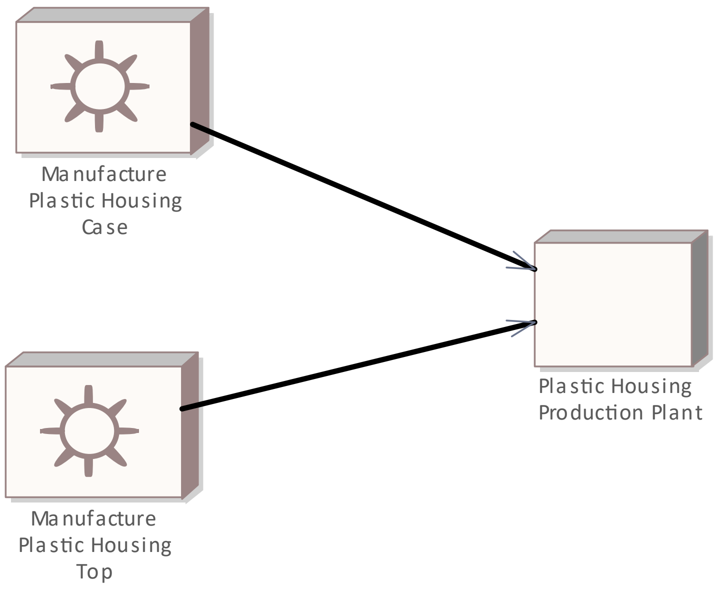

Manufacture Plastic Housing Case
Plastic Housing Production Plant
Manufacture Plastic Housing Top

Figure 8.5: SIPOC model of the plastic housing use case

Moreover, the suppliers deliver goods, information, or energy, and the customers are provided with the outcome of the process. The critical aspect of the model is to recognize system boundaries and delimitations. Depending on the abstraction level, the context diagram indicates different results. In this context, as seen in Figure 8.5, the process is solely represented by the Fischertechnik plant, which is provided with the manufacturing processes for the plastic housing case as the plastic housing top.

After specifying the SIPOC model, the black-box of the process gets into focus. Thereby all processes, including business or manufacturing processes, are modeled. This counts the same for already existing processes of the current Brownfield as well as for target processes. Thus, business processes are modeled with the help of BPMN while manufacturing processes are modeled by utilizing the value stream mapping method (Rother and Shook, 2003). Again, these processes are more or less granular depending on the hierarchy level. In this case, concerning the Fischertechnik plant, such processes might be "produce plastic housing" or "order receipt". As the case study does not have business processes specified due to mainly being located at lower hierarchy levels, now examples for such a diagram are given in this context. Nevertheless, the Business Layer's last part is the requirements specification with SysML. Thereby, the requirements and their dependencies are shown in detail. As this engineering process aims to transform the system from the production of cylinder heads towards plastic housings, a particular focus is set on elaborating requirements for this transformation. The results are thereby shown in Figure 8.6.

The image is split into two different kinds of requirements. The ones on the left address the individual production of plastic housings, while the ones on the right are derived from general business goals. In more detail, to individually produce the plastic housings, the punching station should be informed of new arrivals and consider any house originating from any direction of the assembly line. Thereby, the punching station also needs to have specifications about the holes to be punched. All manufacturing data should be collected to address the company's KPIs. Higher-level requirements specify that new production equipment shall be flexibly deployed, and the Fischertechnik plant shall ensure the configuration of different products. Thereby a wide variety of products with small lot sizes should be enabled by the punching station, and new product variants should be introduced without lengthy interruptions of the production process.

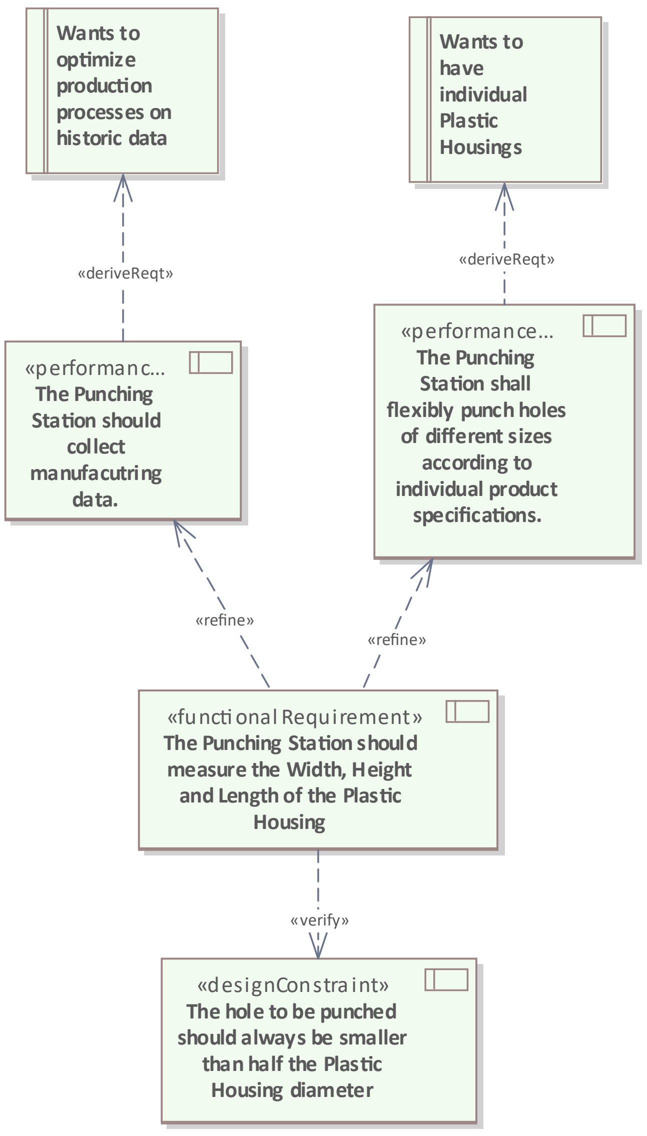

Wants to optimize production processes on historic data
Wants to have individual Plastic Housings
«deriveReqt»
«deriveReqt»
«performanc... The Punching Station should collect manufacutring data.
«performance... The Punching Station shall flexibly punch holes of different sizes according to individual product specifications.
«refine»
«refine»
«functional Requirement» The Punching Station should measure the Width, Height and Length of the Plastic Housing
«verify»
«designConstraint» The hole to be punched should always be smaller than half the Plastic Housing diameter

Figure 8.6: Requirement model of the plastic housing use case

Function Layer

The next step introduces the specification of all used system functions. This includes modeling the functions as black- and white-box with all in- and outputs. In the functional architecture, the interconnection between the functions is depicted, while the white-box shows how the input is translated into the output. Additionally, interferences and disturbances are shown in this model of the existing Fischertechnik industrial plant.

This distinction allows the classification of internal or external faults. Examples of such functions are milling, assembling, or grinding. The milling function is thereby shown in Figure 8.7. The input of these functions is the plastic housing specifications, while the output is a transport job for the crane and a clean job. Interferences could be problems with energy supply or temperature variances, while disturbances are physical issues or deviations in specifications.

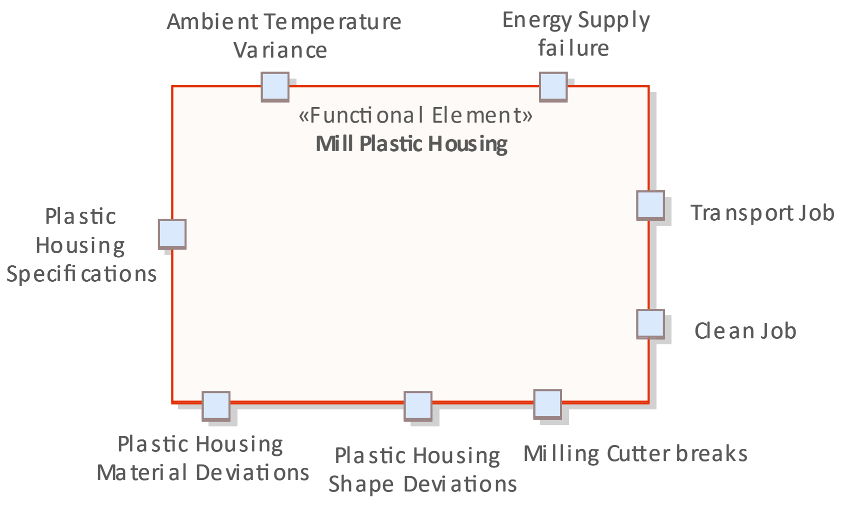

Ambient Temperature Variance
Energy Supply failure
«Functional Element»
Mill Plastic Housing
Plastic Housing Specifications
Transport Job
Clean Job
Plastic Housing Material Deviations
Plastic Housing Shape Deviations
Milling Cutter breaks

Figure 8.7: Milling function of the plastic housing use case

The milling function shows an exemplary implementation of any function to be fulfilled within the system. While functions for Greenfield systems are elaborated differently, this is subsequently outlined in the next section.

Integration Layer

The next step deals with connecting the functions to their logical counterparts. These logical system components carry out those functions and implement the logical system architecture. In this example, the Fischertechnik system of a milling station that fulfills the milling function exists. As visible in Figure 8.8, the milling machine is composed of an Assembly Line 24V, a Milling Machine Fischertechnik education, and a Roboter Niryo One at the Asset Layer of the case study. However, within the Integration Layer, the milling station could be monitored using a SCADA-system, while two interfaces deal with exporting data to external units. In more detail, an HMI is interconnected via an ethernet interface, while a near-field communication (NFC) chip reader receives data via an NFC interface.

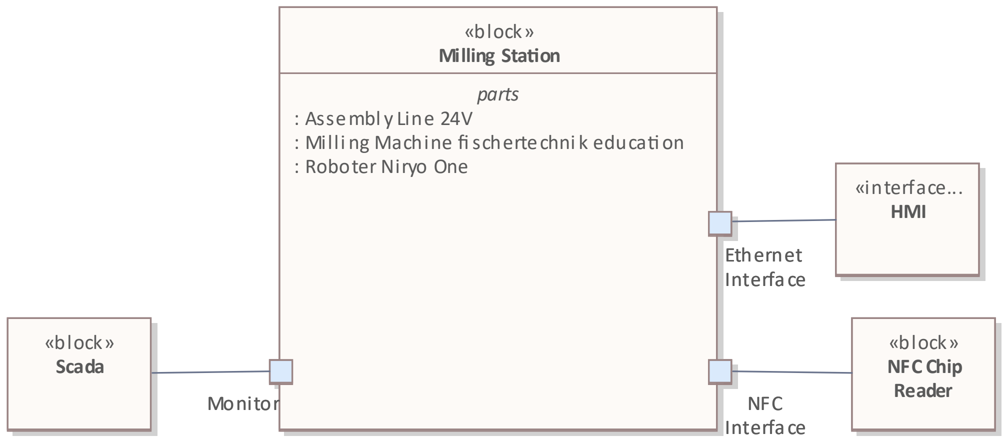


Figure 8.8: Integration Layer diagram of the plastic housing use case

Those logical components are modeled with different modeling languages from UML and SysML. The Integration Layer can be seen as this logical architecture and does not specify any concrete implementation. If any system component already exists and ought to fulfill a particular function, the Integration Layer might be surpassed, and functions are directly traced to the Asset Layer.

Asset Layer

At last, the technical representation of the physical component is modeled by integrating all interfaces and transforming physical aspects or events into virtually treatable objects. These objects could be used in the upper four layers of RAMI 4.0 and are transformed in the Integration Layer. For example, physical interfaces of technical components are translated into virtual components, and events are translated into information objects, which is the primary source for converting the physical system to its virtual representation. Within the Fischertechnik case study, as seen in Figure 8.9, the model of the Asset Layer contains all technical elements, like 3D printers, a bypass assembly line, a milling, and a rotation station as well as a gantry crane.

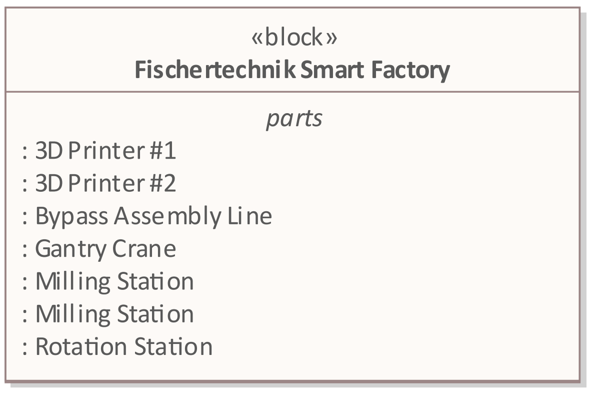


Figure 8.9: Asset Layer diagram of the plastic housing use case

In this layer, all needed information for enabling RTE with AutomationML is stored within the SysML blocks. Therefore, the Asset Layer is the basis for other engineering phases, either on different abstraction levels of the modeling framework or with other engineering tools apart from RAMI 4.0. Thus, at the lower granularity levels, the integration of the punching station is addressed. The guideline for model-driven engineering of a Greenfield is applied, as mentioned in the next section.

8.2.2 Model-driven Engineering of a Greenfield

As aforementioned, systems engineering has different possibilities with the RAMI 4.0 modeling framework. While model-driven engineering of a Brownfield considers already existing systems, model-driven engineering of a Greenfield allows to apply of the framework to an entirely new playground. On the other hand, this means that extensive methods must be utilized to develop parts or single aspects of the system. Those methods are explained in detail in the following paragraphs and the case study context. Hence, this case study aims to make the production of plastic housings more flexible. In smaller quantities, the transformation is executed with the help of the architecture modeling framework. In more detail, after previously describing the architecture of the currently used plant, the TO-BE architecture is specified. As the description of the whole case study architecture would exceed the scope of this evaluation strategy and could be transferred to any other system, a particular focus is set on transforming the punching station. This mainly takes place at the workstation level of the architectural matrix. To describe the comprehensive use case, all models for all engineering

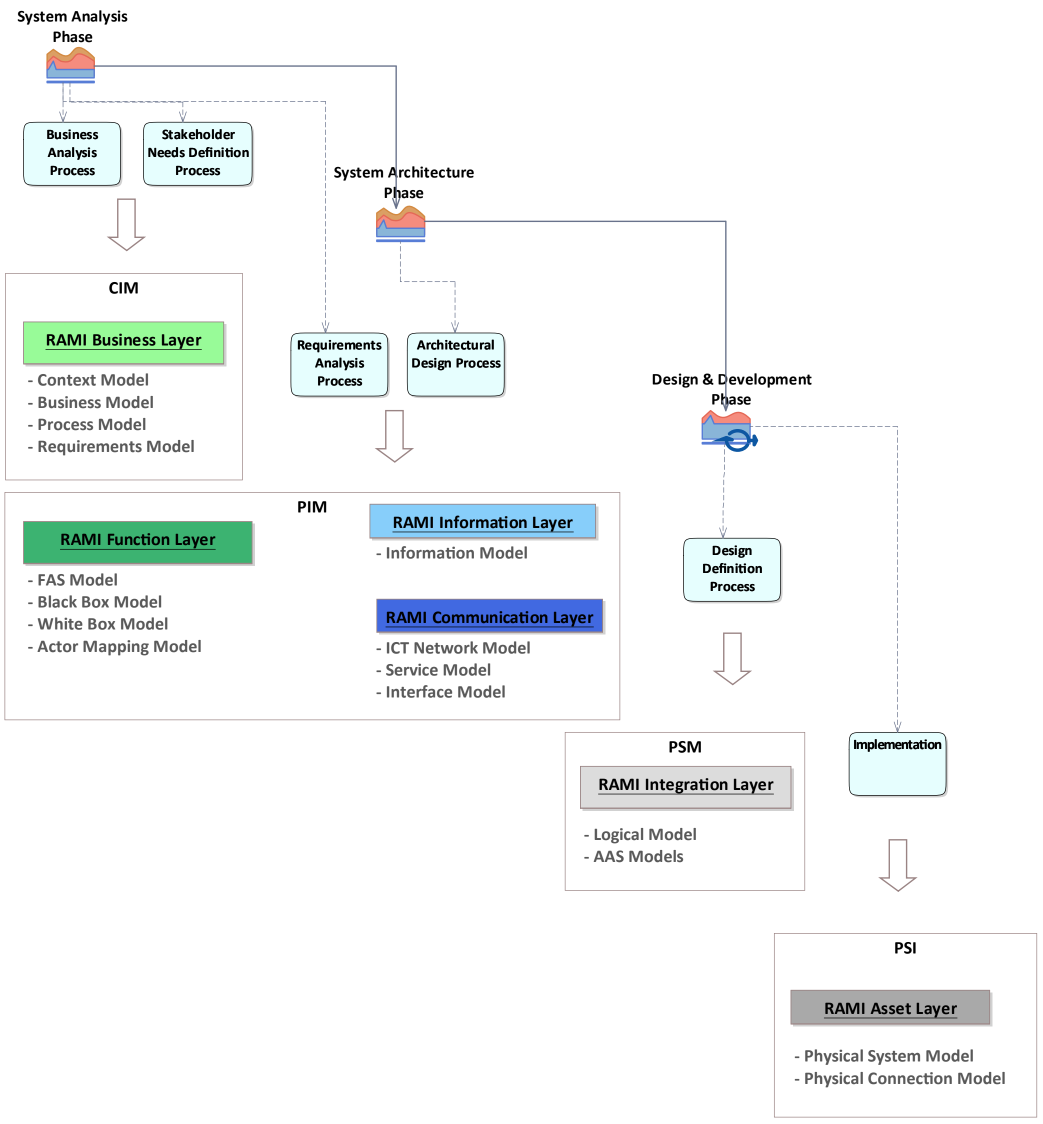


Figure 8.10: Applied process and models for Fischertechnik use case

views are utilized, just as delineated in Figure 5.6, which have been refined in Figure 8.10. All used models are explained in detail in the respective subsections.

Work Station Business Layer

Similar to the description of the AS-IS architecture, the first step is the specification of the system context at this level. This means the punching station system is the SoI, which has several in- and outputs. As demonstrated in Figure 8.11, the plastic housing delivers its specifications on manufacturing, while a sensor provides current measurements. The robot thereby gives the plastic housing and also grabs the final product. Finally, after the punching station is finished punching the plastic housing, it will be returned to the robot, which further processes the punching station. The context diagram is again modeled with the SIPOC elements.

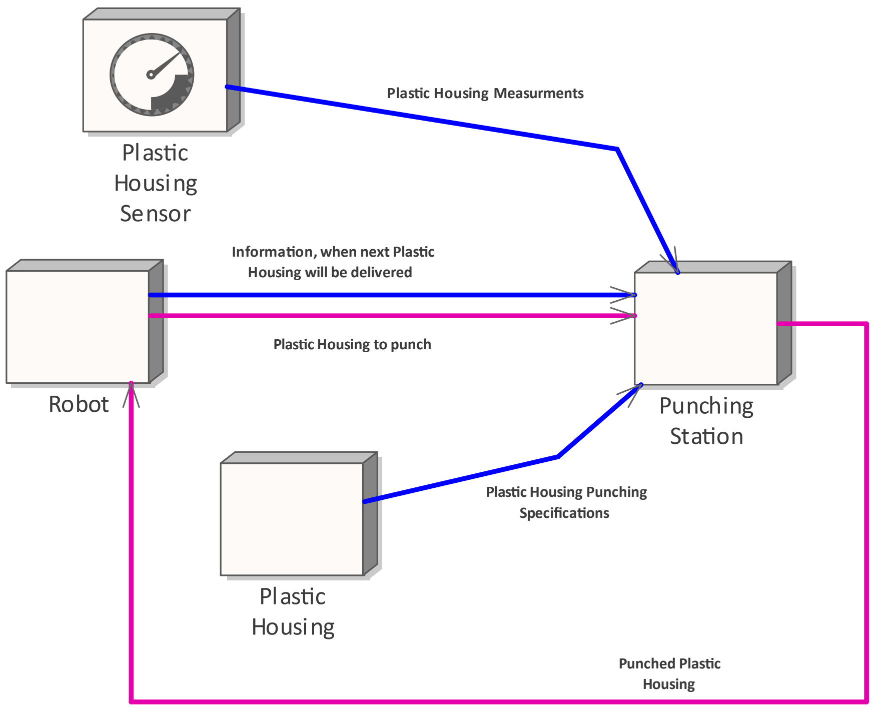

Plastic Housing Sensor
Plastic Housing Measurments
Information, when next Plastic Housing will be delivered
Robot
Plastic Housing to punch
Punching Station
Plastic Housing Punching Specifications
Plastic Housing
Punched Plastic Housing

Figure 8.11: Context model of the Fischertechnik use case

Next, the second step specifies all business and manufacturing processes, as depicted in Figure 8.12. In more detail, a scenario diagram shows the interactions of users with the system as well as the elaboration of functional requirements. Resulting of this and modeled with an UML use case diagram, business cases or high-level use cases exist, which show the intended functionality of the SoI. Additionally, in this model, the stakeholders and their interaction with the system are shown as high-level use cases. In this case study scenario, four different use cases have been elaborated. Thus, the production engineering department interconnects with the punch hole according to the specifications use case, which is extended by the measure plastic housing use case. However, before punching the plastic housing, information about the spot to be punched needs to be gained, which also counts for the specifications of the plastic housing itself. Those specifications are derived from the customer's needs or inserted via a product configurator. When comparing these processes to the ones mentioned in the Brownfield architecture, it can be recognized that they are located in several hierarchy levels below.

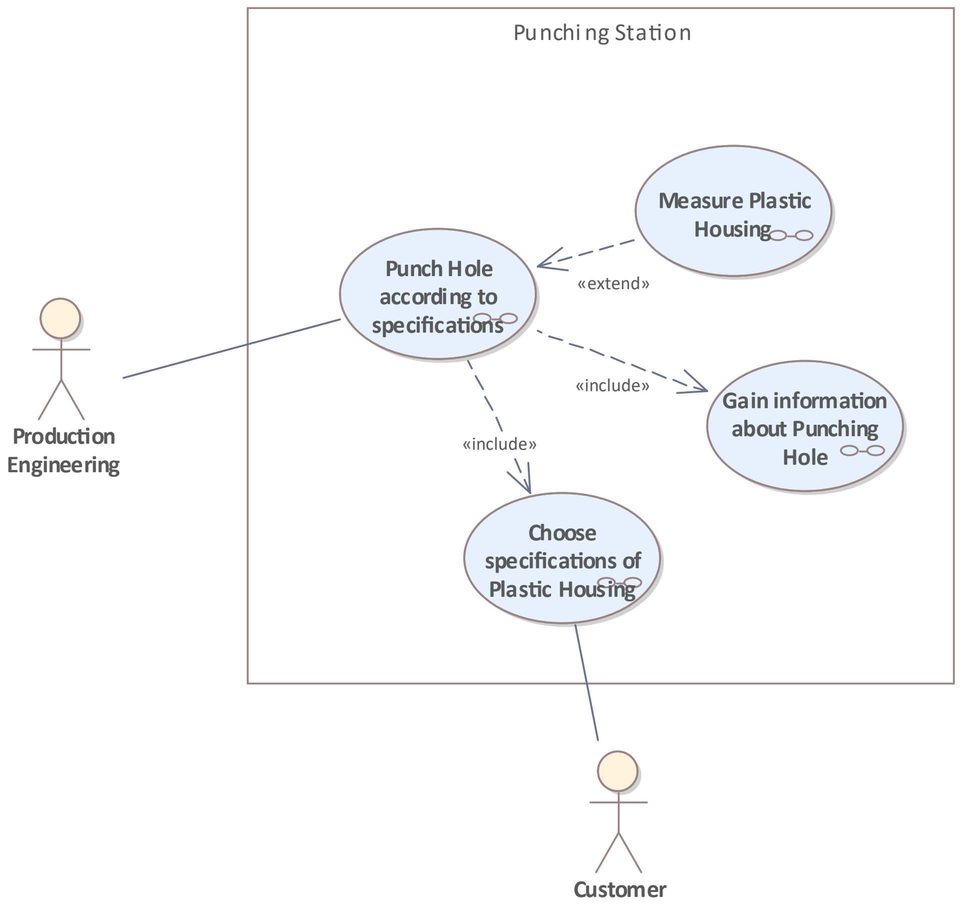

Punching Station
Measure Plastic Housing
Punch Hole according to specifications
«extend»
«include»
Gain information about Punching Hole
«include»
Choose specifications of Plastic Housing
Production Engineering
Customer

Figure 8.12: Processes of the Fischertechnik use case

After specifying the use cases, all already existing system processes are outlined in detail, which helps recognize weaknesses or identify optimization potential to address in the TO-BE system. For describing the business processes, BPMN is used. In contrast, the manufacturing processes are described with the value stream mapping method, which is implemented as domain-specific language in the DSL. The previously mentioned processes would be examples of business processes, while the actual manufacturing of the plastic housing is a typical example of a manufacturing process. Using the value stream mapping method, external sources deliver anything that needs to be delivered. Each process step is represented by a particular symbol, while their interaction could be manual streaming, electronic streaming, or simply pushing forward materials.

At last, after defining the system context, all stakeholders, processes, and use cases of the system, the resulting requirements are engineered. Therefore, the modeling framework uses SysML and allows users to model requirements with different notations. In this case, requirements for the punching station are defined in Figure 8.13, like bi-directional assembly line movement or storage of manufacturing information within the plastic housing.

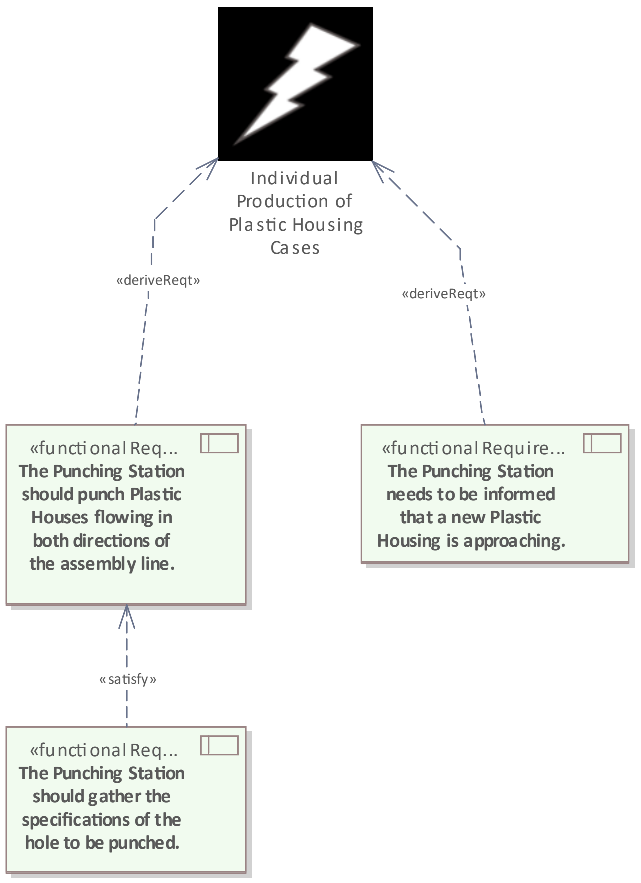

Individual Production of Plastic Housing Cases
«deriveReqt»
«deriveReqt»
«functional Req...
The Punching Station should punch Plastic Houses flowing in both directions of the assembly line.
«functional Require...
The Punching Station needs to be informed that a new Plastic Housing is approaching.
«satisfy»
«functional Req...
The Punching Station should gather the specifications of the hole to be punched.

Figure 8.13: Requirement model of the Fischertechnik use case

Work Station Function Layer

In the next step, the Function Layer is modeled in detail by refining the scenario model. If already used functions are refined, they are modeled with a specific DSL, utilizing functional elements to specify inputs, outputs, disturbances, and interference. However, suppose particular use cases or process steps are not yet realized by a function, which should usually be the case for modeling a Greenfield. In that case, new functions need to be elaborated. In this case, the FAS method (Weilkiens et al., 2015) has been integrated into the modeling framework, which appears to be a state-of-the-art method for elaborating new functions. The previously defined use cases are specified with activity diagrams to give small insights into this method, which show their intended functionality. An activity diagram might have any number of processes and hierarchy levels. However, each granular process needs to be modeled as an action. After modeling all activity diagrams to all intended processes, according to the FAS method (Weilkens et al., 2015), actions fulfilling similar tasks are thereby grouped, as visible in Figure 8.14. Those actions are then summarized and dealt with as a foundation for the system's functions.

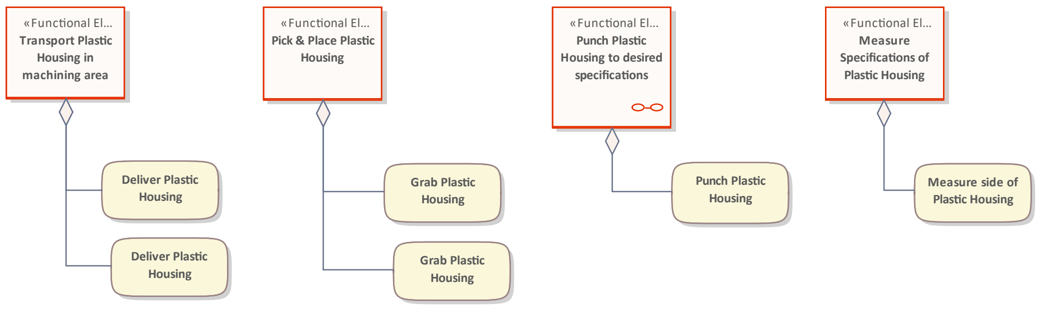

«Functional El...
Transport Plastic Housing in machining area
Deliver Plastic Housing
Deliver Plastic Housing
«Functional El...
Pick & Place Plastic Housing
Grab Plastic Housing
Grab Plastic Housing
«Functional El...
Punch Plastic Housing to desired specifications
Punch Plastic Housing
«Functional El...
Measure Specifications of Plastic Housing
Measure side of Plastic Housing

Figure 8.14: Functional grouping of the Fischertechnik use case

In the context of the case study, examples of such actions are given with the delivery of the plastic housing, grabbing the plastic housing, punching it, and measuring the dimensions of the plastic housing. Those might then be summarized as the functions transport plastic housing in the machining area, pick & place plastic housing, punch plastic housing to desired specifications, and measure the specifications of the plastic housing. Those functions are the basis for defining the system's functional architecture, representing the functionalities the physical components need to fulfill. To give more detailed insights, the single functions are additionally refined.

After each function has been elaborated on and specified, a more detailed representation of the function is given. Thus, every function is defined as a black- or white-box. This model is implemented with specific DSL diagrams consisting of function objects, ports, and connections. Additionally, the single functions are specified as black- and white-box models with in-, outputs, disturbances, and interferences by utilizing SysML block definition and internal block diagrams. The black-box diagram thereby shows the external dependencies of the function, while the white-box illustrates how the input is transferred to the output. By using internal block diagrams, the element securing this process is defined as the function's properties. Thus, when thinking further, those properties might again be functions on a lower granularity level.

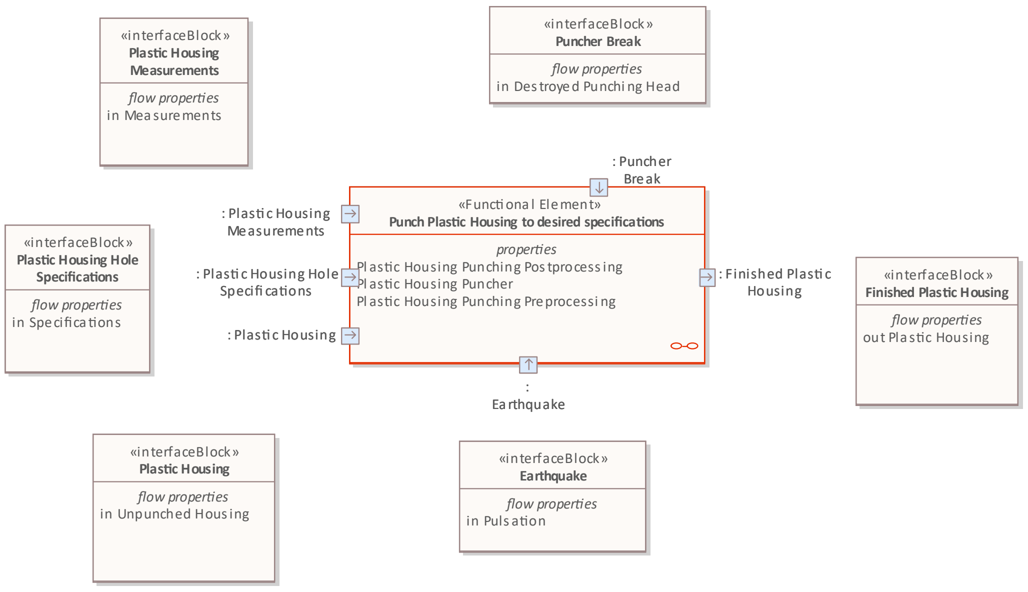


Figure 8.15: Black- & white-box function of the Fischertechnik use case

For a specific example, one function within the case study might be punching the plastic housing to desired specifications, as seen in Figure 8.15. The image shows all inputs, outputs, disturbances, interference, and properties of the mentioned function. All external elements are thereby represented as external interfaces, which are aligned to the function's ports. In this specific case study, the inputs of the function are represented by the measurements of the plastic housing and its specifications for punching the hole. The plastic housing is an input to fulfill the function, representing its export when the whole is punched. A typical disturbance to hinder the function's proper execution will be a malfunction of the puncher. However, the properties to punch the plastic housing according to the desired specifications are given within the internal block diagram. In more detail, the punching process needs pre-processing and post-processing, while the punching function is applied to manufacture the plastic housing.

Subsequently, the functional architecture could be developed after specifying each system's functions in detail with black-box and white-box diagrams. The elaborated functions and their interconnection within this model are visualized to consider eventual correlations, shown in Figure 8.16. Doing so compares inputs and outputs, and the information exchange between their ports is modeled. This means the output of other functions could fulfill some inputs of specific functions. Within this model, those in- and outputs are compared and interconnected with information object flow relations. The functional architecture indicates the information flow and builds a base for the RAMI 4.0 Information Layer.

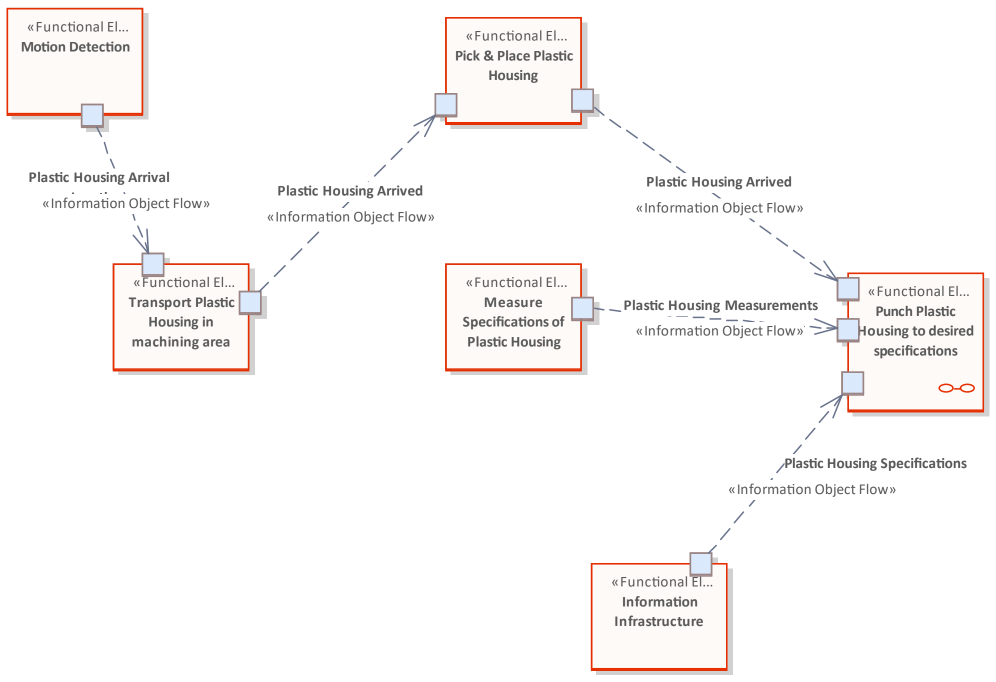

«Functional El...
Motion Detection
«Functional El...
Pick & Place Plastic Housing
Plastic Housing Arrival
«Information Object Flow»
Plastic Housing Arrived
«Information Object Flow»
Plastic Housing Arrived
«Information Object Flow»
«Functional El...
Transport Plastic Housing in machining area
«Functional El...
Measure Specifications of Plastic Housing
Plastic Housing Measurements
«Information Object Flow»
«Functional El...
Punch Plastic Housing to desired specifications
Plastic Housing Specifications
«Information Object Flow»
«Functional El...
Information Infrastructure

Figure 8.16: Functional architecture of the Fischertechnik use case

For example, the previously mentioned functions are interconnected with those not shown in the context of the FAS method. The first function starts the whole process and is given with motion detection. Suppose the punching station detects any motion.

In that case, the arrival of the plastic housing is passed to the transport function, which deals with transporting the plastic housing to the machining area. Additionally, before punching the plastic housing to the specifications, a pick & place function deals with informing the punching function that new plastic housing is arriving. Moreover, the measurements of the plastic housing are scanned, and the plastic housing itself passes the specifications with the help of an information infrastructure.

Work Station Information Layer

After modeling the communication architecture of the elements and their interconnections, the information exchange between the functions is depicted. During model-driven engineering, no existing components are yet given, and the Information Layer should help find components that can fulfill the functions. This model used the current processes and interconnection of the functions and summarized them to information objects that might be implemented for mutual access. Thereby, the Information Layer inherits a vital role, as AI-based optimizations or data evaluations could be made based on its outcome.

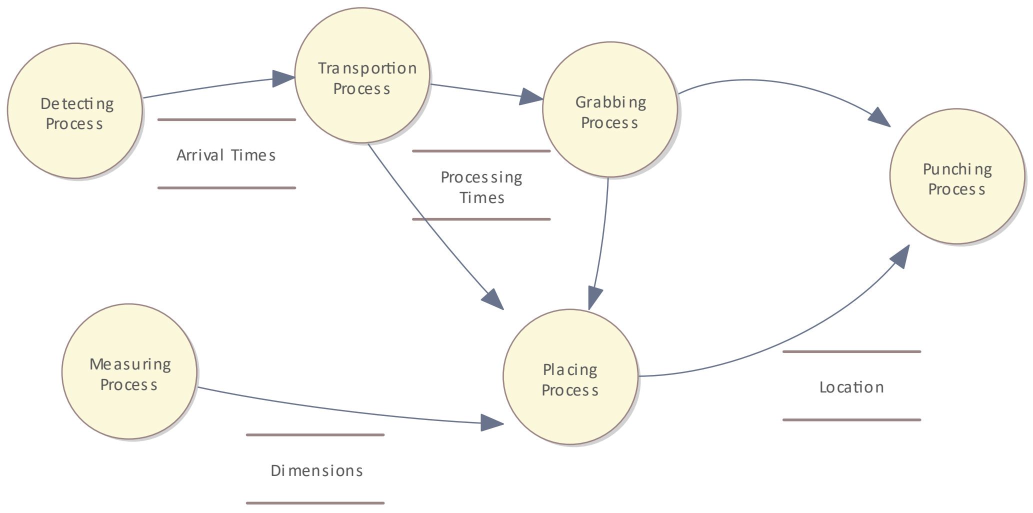


Figure 8.17: Information Layer of the Fischertechnik use case

However, a more detailed view of the function interrelations is given to support the systems engineering process. Functions are represented as processes and exchanged data is modeled as a data store. Different processes, like detecting, measuring, transporting, grabbing, placing, and punching, have been elaborated in the case study context. The data to be stored is represented as Arrival Times, Processing Times, Dimensions, and Location of the plastic housing. As this layer does not offer a lot of information for actually choosing logical and physical components to fulfill the functions, results from system optimization could play essential roles. Such optimization potential could be the alignment and the sequence of order to process the functions or AI-based safety and security by design.

Work Station Communication Layer

Based on the data exchange, it becomes evident that different functions need to provide specific data while other functions consume this data. While the data exchange is modeled in the Information Layer, the interfaces are part of the Communication Layer. Thus, when executing model-driven engineering with the RAMI 4.0 modeling framework, the next step is to define the interfaces between the functions, including technologies to transmit the data. As RAMI 4.0 itself represents a SoA, based on the respective interface, it is elaborated whether a service is provided or it is consumed, which can be exemplarily seen in Figure 8.18. The services should consider the data flow and provide them via in- or output interfaces. Those interfaces are described as request points or service points, respectively. Request points deal with consuming a service by needing specific information, while service points provide this information. Thus, as illustrated in the image, each interface is traced to a service interface providing or requesting information. In more detail, those interfaces expose some capability of the functions or system components, depending on which element is realizing the service. However, those capabilities satisfy particular requirements elaborated within the Business Layer. Based on this architecture, the components to fulfill those services might subsequently be defined in the bottom layers of RAMI 4.0.

In the case study context, one specific example is delineated, which gives an idea of how to deal with all other interfaces. This example uses the interface of NFC, which provides measurements and specifications for the punching process. Thus the punching process owns a measurement & specification request point that asks for this detailed information via NFC. By doing so, the system inherits the capability to individually punch the plastic housing according to the specifications needed, which fulfills several requirements. In more detail, the measurement of the dimensions of the plastic housing and the gathering of the specifications. Furthermore, on different granularity levels, this capability enables flexible production in lot size 1.

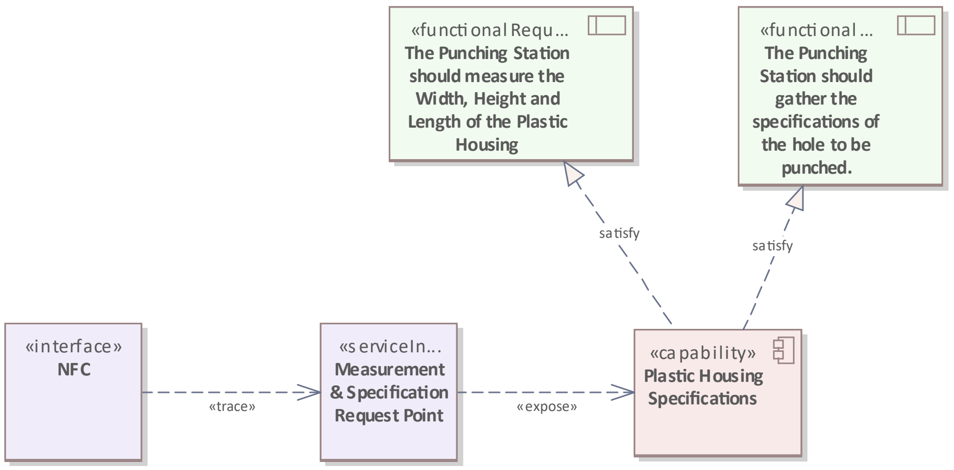

«functional Requ...
The Punching Station should measure the Width, Height and Length of the Plastic Housing
«functional ...
The Punching Station should gather the specifications of the hole to be punched.
satisfy
satisfy
«interface»
NFC
«serviceIn...
Measurement & Specification Request Point
«capability»
Plastic Housing Specifications
«trace»
«expose»

Figure 8.18: Communication Layer of the Fischertechnik use case

Work Station Integration Layer

The RAMI 4.0 Integration Layer deals with, as the name assumes, the integration of the physical assets into the virtual world. Thus, it could be considered logical architecture inheriting components which are not part of the existing production system but deal with information virtualization. If needed, this layer defines logical components of the system, which can address all previously elaborated aspects. Thus, requirements need to be considered and fulfilled by particular components, while the execution of functions is also a significant aspect. Moreover, the data exchange via information objects has to be realized, and the specified services are provided or requested by single logical system components. Suppose physical assets already exist which can fulfill the required needs. In that case, the Integration Layer could be passed, and the system components could be directly specified within the Asset Layer. However, different solutions for individual needs might be possible in most cases. In this case, elaborating on the Integration Layer is essential to model-driven engineering. By specifying the logical architecture, 150%-solutions could be defined, and different variants or multiple proposed solutions could be offered. As shown in Figure 8.19, the Integration Layer thus could play a significant role in the process if the system is developed from scratch.

As seen in Figure 8.19, the logical architecture of the case study consists of several elements. While the punching station is the SoI within the hierarchy level of the workstation, its logical architecture consists of elaborated components that can fulfill the functions, requirements, and information exchange. In more detail, a robot could take over picking and placing tasks while the punching machine deals with punching the plastic housing. Furthermore, the assembly line deals with transporting the plastic housing, while a measurement sensor reads the specification and a motion actor considers the arrival of new plastic housings. The communication module offers the possibility to exchange data between the components.

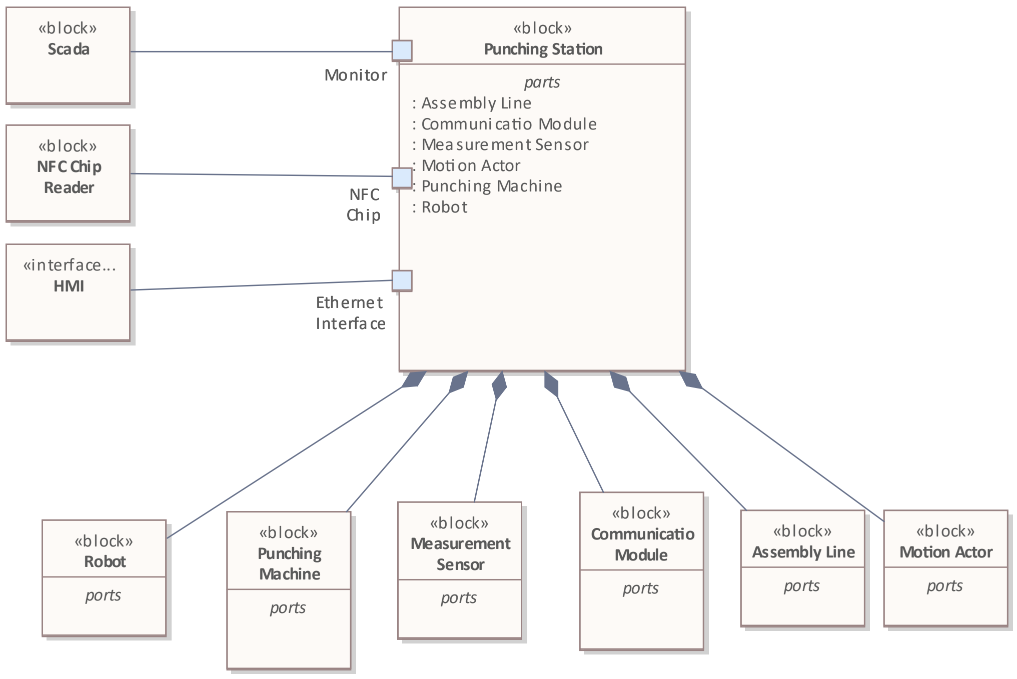


Figure 8.19: Integration Layer of the Fischertechnik use case

As additionally visible, the interfaces or events are translated within the Integration Layer of RAMI 4.0. Thereby, either existing ones are used, or new ones are created. In this scenario, the punching station is controlled by a SCADA system and a PLC. Additionally, technicians might control them via an HMI. This allows to integrate the logical elements into the virtual world and find physical components that realize those elements, including their interfaces.

Work Station Asset Layer

As mentioned, the following steps deal with fulfilling the specified functions or logical components with actual components. This is where the AS-IS and TO-BE architecture are amalgamating, as the original components of utilizable manufacturing units already exist. Thus, this step indicates which functions are fulfilled by which components and, the other way round, which component executes which function. This shows whether new manufacturing units need to be developed or existing ones could be used. As this case study aims to transform a current system, components might be re-used from the original production network. Therefore, it must be elaborated if new components need to be created. In the case study, as seen in Figure 8.20, some assembly lines can be used while new ones need to be set up.

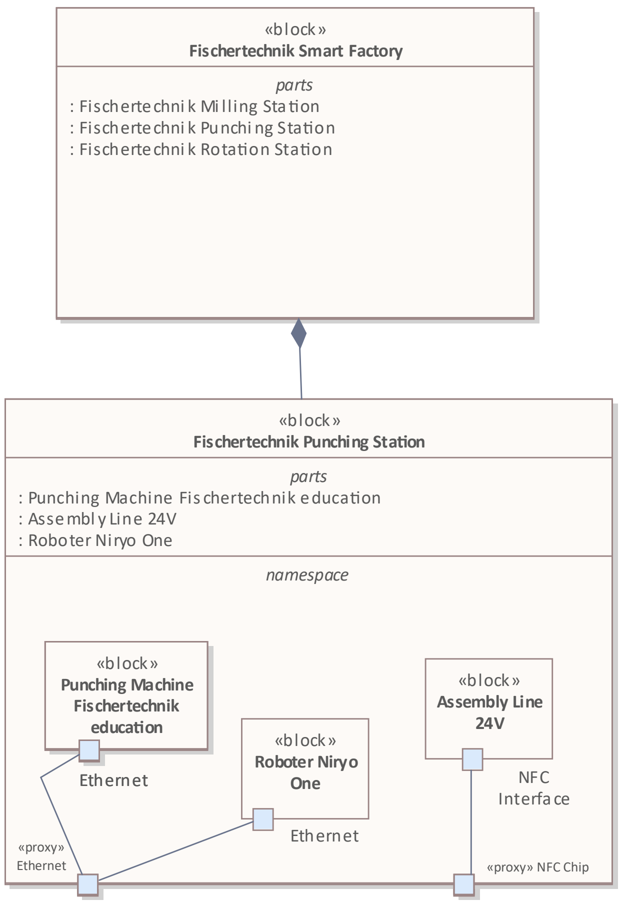

«block»
Fischertechnik Smart Factory
parts
: Fischertechnik Milling Station
: Fischertechnik Punching Station
: Fischertechnik Rotation Station
«block»
Fischertechnik Punching Station
parts
: Punching Machine Fischertechnik education
: Assembly Line 24V
: Roboter Niryo One
namespace
«block»
Punching Machine Fischertechnik education
Ethernet
«block»
Roboter Niryo One
Ethernet
«block»
Assembly Line 24V
NFC Interface
«proxy» Ethernet
«proxy» NFC Chip

Figure 8.20: Asset Layer of the Fischertechnik use case

Moreover, a new robot needs to be set up, while the gantry crane could be used as it is. The same counts for milling and grinding stations, while the punching station is newly established. The following steps are similar to the ones of the AS-IS architecture. This means the Asset Layer of RAMI 4.0 contains several physical elements, which are partially visualized. It can be seen that the Fischertechnik Smart Factory has a Fischertechnik Punching Station. This station consists of three separate components. To realize the punching functionality, the punching machine Fischertechnik education is utilized. The logical component of the robot is recognized by a robot niryo one, while a conventional assembly line 24V is used to deal with the transporting functionality. Those components are interconnected with physical interfaces, which might be virtualized within the Integration Layer. However, iterating the layers from right to left deals with digital twin development, which is outlined in more detail in the next section.

The system's alignment with the matrix enables the flexible manufacturing system's specification and the components' treatment. According to the architecture method of TOGAF, after implementing the system about the TO-BE architecture, the architecture described in this section becomes the new AS-IS architecture, and unique aspects are elaborated within the next iteration step of TOGAF with another TO-BE system architecture. This means that both concepts are usually arranged in connection with each other but could also stand uniquely. Thus, within the next iteration, a Greenfield turns into a Brownfield, while a Brownfield remains a Brownfield.

8.2.3 Digital Twin Development

As already mentioned, the RAMI 4.0 modeling framework also allows the possibility to specify digital twins of already implemented systems. This is done by iterating the layers from bottom to up, beginning with the Asset Layer and finishing with the Business Layer. This is the counterpart to model-driven engineering, primarily using an already existing system architecture. The aspects addressed to enable the virtualization are iterated from bottom to up. This means the solution area is described first, and the problem area is based on used technologies. The order of applied engineering steps and used models for this use case is outlined in Figure 8.21 and further delineated in the remainder of this section.

To give a small outlook, the digital twin development is initiated by specifying the Asset Layer of the system. In the context of this case study, components like the milling or grinding machine, as well as the gantry crane, are used. Thereby, all physical interfaces and their connections are added to the model. As the crane might not be able to

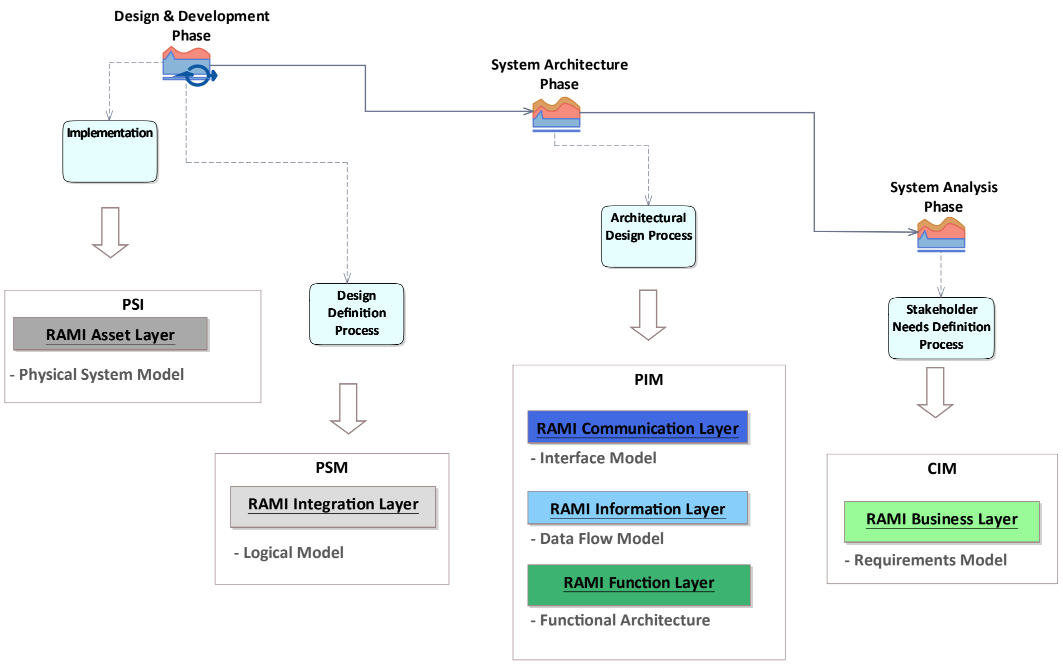


Figure 8.21: Applied process and models for Digital Twin Modeling

communicate independently, events need to be virtualized within the Function Layer. This means all kinds of data are stored in various databases and made available via a server on the one hand, or additional sensors are recording data and providing them to other components. This results in having physical or virtual interfaces. However, those are treated equally in the upper layer of RAMI 4.0, and no particular distinction is made. Thus after modeling either the Asset or Integration Layer of RAMI 4.0, the next step of the digital twin development process is to specify the Communication Layer.

In this scenario, communication with other system components is done with the help of a NFC chip. This allows to subsequently define the ICT infrastructure in the Communication Layer and the Information Exchange in the Information Layer. Thereby, the already existing models are adapted or extended with the new aspects of the system to be transformed. In this case study, those diagrams are taken from the AS-IS architecture, and the interconnection and data exchange to new system components like the robot, the punching station, or the bypassing assembly line have been specified. In other cases, if the TO-BE architecture shows a significant distinction from the currently used system, those models must be created from scratch.

The following model is located within the Communication Layer and outlines the physical interconnection between the components. Modeling interfaces, ports, and protocols support the exchange of information in the system. The resulting network infrastructure and technologies, like Ethernet or connections to the PLC in this scenario, are also part of this model in Figure 8.22.

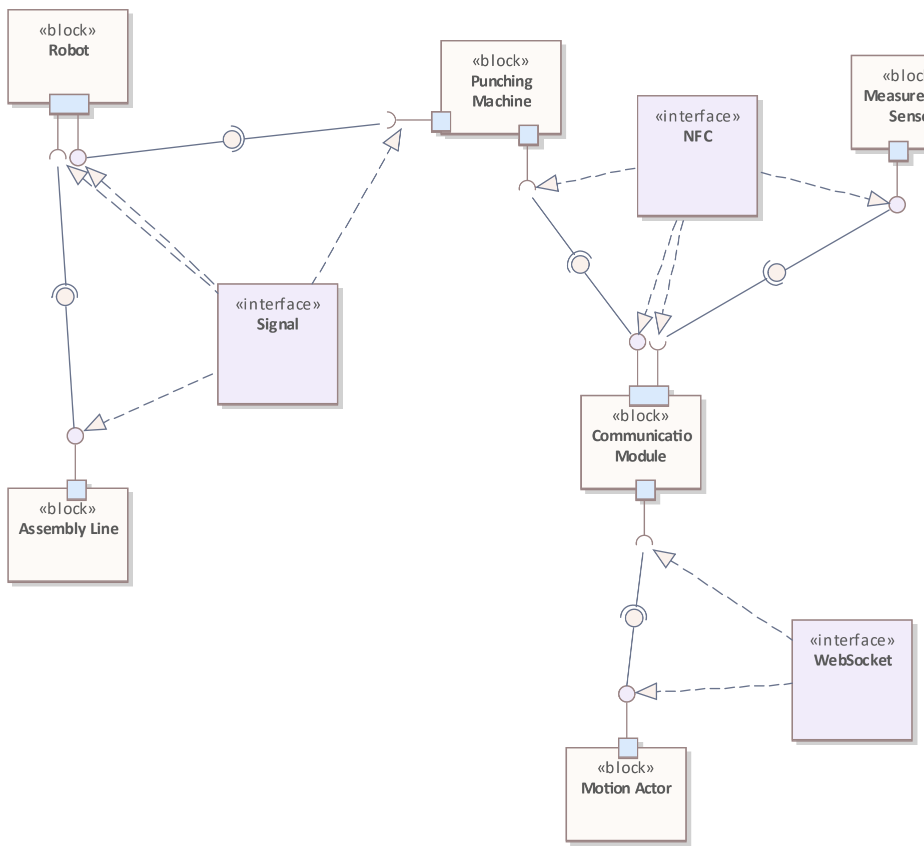


Figure 8.22: Digital Twin Modeling Communication Layer

After modeling the communication architecture of the elements and their interconnections, the information exchange between the functions is depicted by elaborating on the single information objects that are exchanged within the system. By modeling this in the Information Layer, data exchange or management can be elaborated in Figure 8.23. SysML is used as a modeling language for the functions, while the information architecture is modeled with a separate DSL as well as the data models.

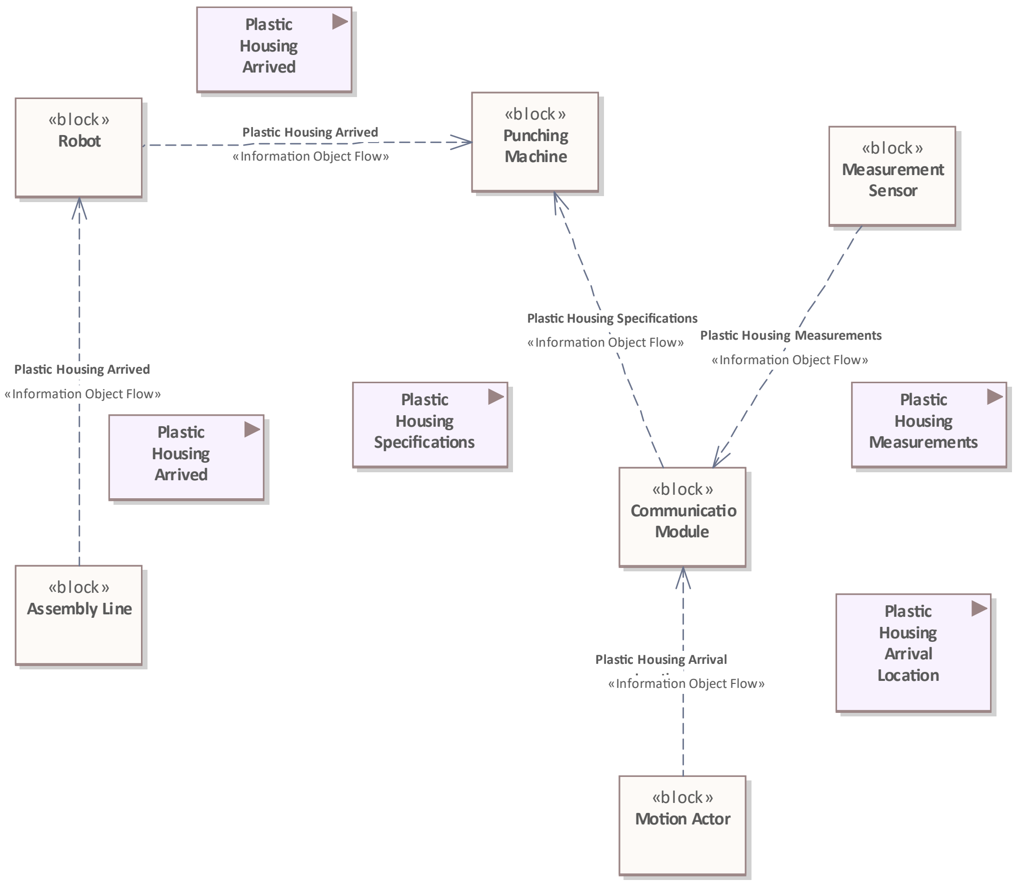

Plastic Housing Arrived
«block» Robot
Plastic Housing Arrived
«Information Object Flow»
«block» Punching Machine
«block» Measurement Sensor
Plastic Housing Specifications
«Information Object Flow»
Plastic Housing Measurements
«Information Object Flow»
Plastic Housing Arrived
«Information Object Flow»
Plastic Housing Arrived
Plastic Housing Specifications
Plastic Housing Measurements
«block» Communicatio Module
«block» Assembly Line
Plastic Housing Arrival
«Information Object Flow»
Plastic Housing Arrival Location
«block» Motion Actor

Figure 8.23: Digital Twin Modeling Information Layer

The specification of the Information Layer results in indicating the data exchange between the components. Thus, in the next step, within the Function Layer, the actual functionality of the components could be modeled. Thereby, physical components are traced to this layer, where their functionality is connected to them. As no new functions need to be elaborated, the FAS method is not applied during the digital twin development. The functions are connected to the already existing system components to ensure traceability. However, if requirements are interconnected with the functions or the components, they are modeled within the Business Layer of RAMI 4.0.

8.3 Exemplary Toolchain Integration of the Modeling Framework

This paragraph evaluates the usability of the RAMI 4.0 modeling framework to be applied within the toolchain. At first, its application of the PPR notation to model systems based on product, process, and resource views is validated. An exemplary PPR model is applied, which falls back on the packaging process case study. This case study makes use of packaging already manufactured goods. This makes the use case ideal to be applied with other manufacturing systems, as it could deal with processing and shipping each product. By applying the use case under the modeling framework, its possibilities for performing RTE tasks are analyzed. This means suitable export and import interfaces are provided, and the utilization of the exported information within other tools in the toolchain is demonstrated. Furthermore, this also allows the evaluation of cross-domain interoperability features of the RAMI 4.0 modeling framework.

8.3.1 PPR Systems Description

Figure 8.24 gives an outlook on how such systems might be modeled within the framework. However, this image solely demonstrates an excerpt of a diagram, all other diagrams of the PPR model follow the same principle and are not addressed in the scope of this thesis. Based on the modeled system, the interconnection between products and processes, as well as processes and resources, might be investigated. To do so, the image demonstrates the packaging process at the highest granularity level. The receipt of manufactured goods initiates this process. Those are transported to the packaging area via a conveyor. At the packaging area, a vacuum gripper crane deals with the transported manufactured goods and processes by handing them over to the packaging station, where the packing process is executed. Next, the packed manufactured good is transported once again with another conveyor. Finally, a gantry crane grabs the goods and places them into the wagon. By doing so, the loading process is executed within the packaging system. After loading the goods, the ultimate state has been reached, and the packaging process is finished.

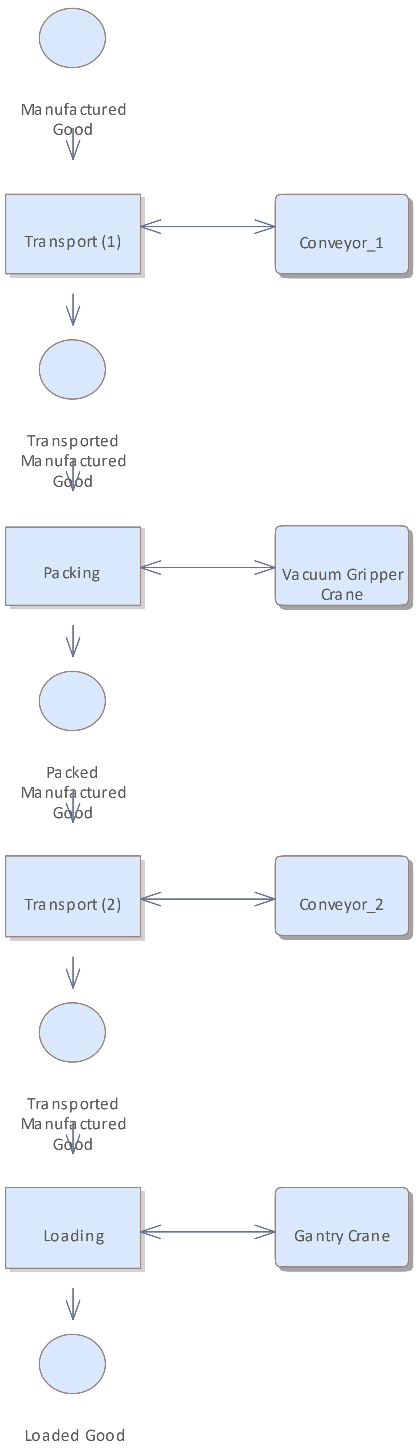
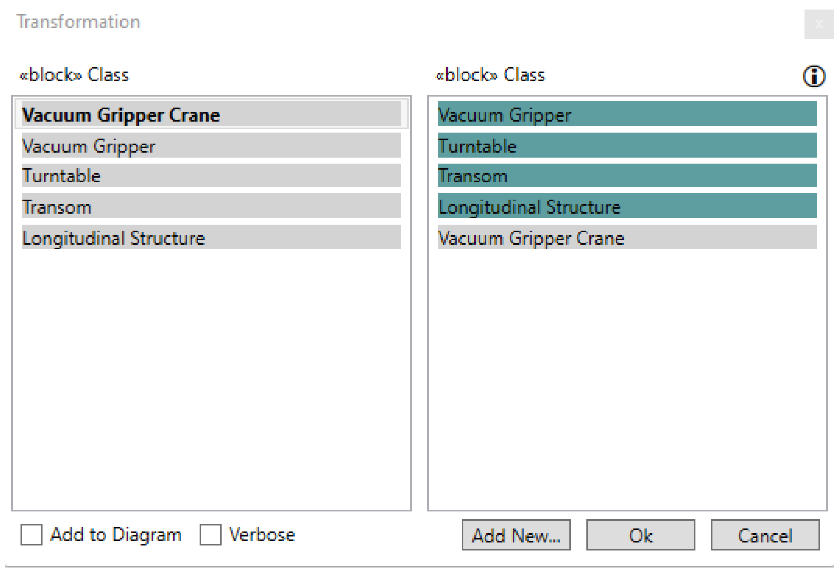


Figure 8.24: PPR model example

The RAMI 4.0 modeling framework additionally offers the possibility to automatically interconnect PPR system elements at different granularity levels. This means the aggregation relation for decomposing the resources might be applied to create such a structure. As connecting the elements from multiple granularity levels is a time-consuming task that must be executed manually, supporting this step would be an essential feature, especially for users without modeling or systems engineering expertise. Thus, the framework contains such tracing functionality if users solely desire to describe a PPR system. The modeling transformation window is extended and offers support in this particular scenario. Therefore, by executing the user interface window, the resource elements of the upper layer on those on the lower layer are listed, as shown in Figure 8.25. In the example of the packaging case study, the vacuum gripper crane is decomposed into a vacuum gripper, a turntable, a transom, and a longitudinal structure. When selecting the crane within the user interface, its sub-components are highlighted as visible with a green background. This indicates that the higher-level component contains a sub-structure. The components not deposited within this sub-structure remain unchanged and are shown with their gray background. On the other hand, when clicking on one element of the sub-structure, the higher-level component on the left-hand side, which represents the selected component's parent, would be highlighted with green background. By executing this window, potential users are provided with high-supportive and automating functionality, as no modeling elements need to be exchanged, or whole diagrams do not need to be compared.

The manual workaround to this function would be the creation of actor mapping diagrams. While this functionality is usually not needed, it has two benefits. At first, it allows to visualize the hierarchical structures of the system within diagrams and thereby exhibit the system's dependencies or complexity. In contrast, the other approaches only allow visualizations with matrices or user interfaces. The second advantage is that manually interconnecting elements leads to fewer errors, as more thinking is involved while expending the manual effort. In contrast, utilizing the GUI only needs two clicks on textual descriptions. Thus, within such a diagram, the elements from the layer above are horizontally aligned. In contrast, the actors from the beyond diagram are horizontally aligned with some space to the upper elements. However, those elements must be manually collected and copied into the actor mapping diagram, representing much manual effort. After copying all the required elements, relations between them could be drawn. Nevertheless, as usually no one-to-one relations are contained within the model, some elements might have multiple lines to other ones. This makes the diagram confusing and hinders investigations of the modeling elements. Thus, by automating this function, usability is strongly enhanced.


Transformation

«block» Class

Vacuum Gripper Crane
Vacuum Gripper
Turntable
Transom
Longitudinal Structure

«block» Class

Vacuum Gripper
Turntable
Transom
Longitudinal Structure
Vacuum Gripper Crane

Figure 8.25: PPR decomposition example

Finally, after describing such PPR systems with the help of the RAMI 4.0 modeling framework, they might be used in external tools for further processing. This could be system optimization by applying simulations or AI, virtual commissioning, or system tests. By doing so, the previously described model and its information need to be available for those other tools within the toolchain. To execute this task, the modeling framework offers different scenarios. Two possible ways to perform this are considered in this context. For example, one possibility is to export the PPR model as XML file, where all needed information is stored. To extract this information, separate transformations are required, which are not elaborated on in the context of this thesis. This option, however, provides certain flexibility, as decisions about extracted attributes could be dynamically determined, and no hard coding of export functionality is needed. On the other hand, the bi-directional interface to AutomationML could also export such a PPR system structure. This option is detailed and delineated based on a different example within the next section.

8.3.2 AutomationML RTE Evaluation

Before actually validating the interface between RAMI 4.0 and AutomationML to enable RTE, the Fischertechnik smart factory needs to be modeled according to the specifications of RAMI 4.0. As delineating the complete model, including all RAMI 4.0 layers, would exceed the scope of this scenario, only the needed aspects to validate the RTE approach are outlined. This means that the Asset Layer, which describes the physical systems as instances, is used for model-to-model transformations. This layer results from previously modeling the other layers, where requirements, functions, or data exchange are dealt with. Finally, real-world systems are defined, which realize each of the mentioned aspects. As those system components need to be used in other engineering tools, the corresponding model is used for RTE. Figure 8.26 shows that the Fischertechnik components are modeled with SysML diagrams. In this case, particular focus is set on the punching station of the smart factory, which is realized with a SysML block within a SysML block definition diagram.

The punching station is part of the complete Fischertechnik smart factory, just like other stations, like milling or a rotation station. Additionally, two different 3D printers are included for printing parts of the plastic housing and an assembly line or a gantry crane, which deals with transporting the plastic housing. At a lower granularity, the punching station itself exists of a punching machine, a separate assembly line as well as a robot. To interconnect with other system components on the RAMI 4.0 Communication Layer, different communication interfaces, like NFC or Ethernet, are available. At the same time, a measurements sensor or a motion sensor creates events from the punching station within the RAMI 4.0 Integration Layer. Finally, the robot contains a motion module and a gripping module.

Next, the first scenario describes the creation of an AutomationML-file based on the developed system model according to RAMI 4.0. The respective SysML diagrams are created within the Asset Layer resulting from MBSE. Subsequently, those models are exported into an AutomationML-file, which is stored in the file system of the operating system. The structure of this file can be viewed in Figure 8.27. When comparing this image to the SysML model depicted in Figure 8.26, it can be seen that the developed system is identical to the exported AutomationML InstanceHierarchy model, where it can be used in other tools within the engineering toolchain, like factory acceptance tests or simulations.

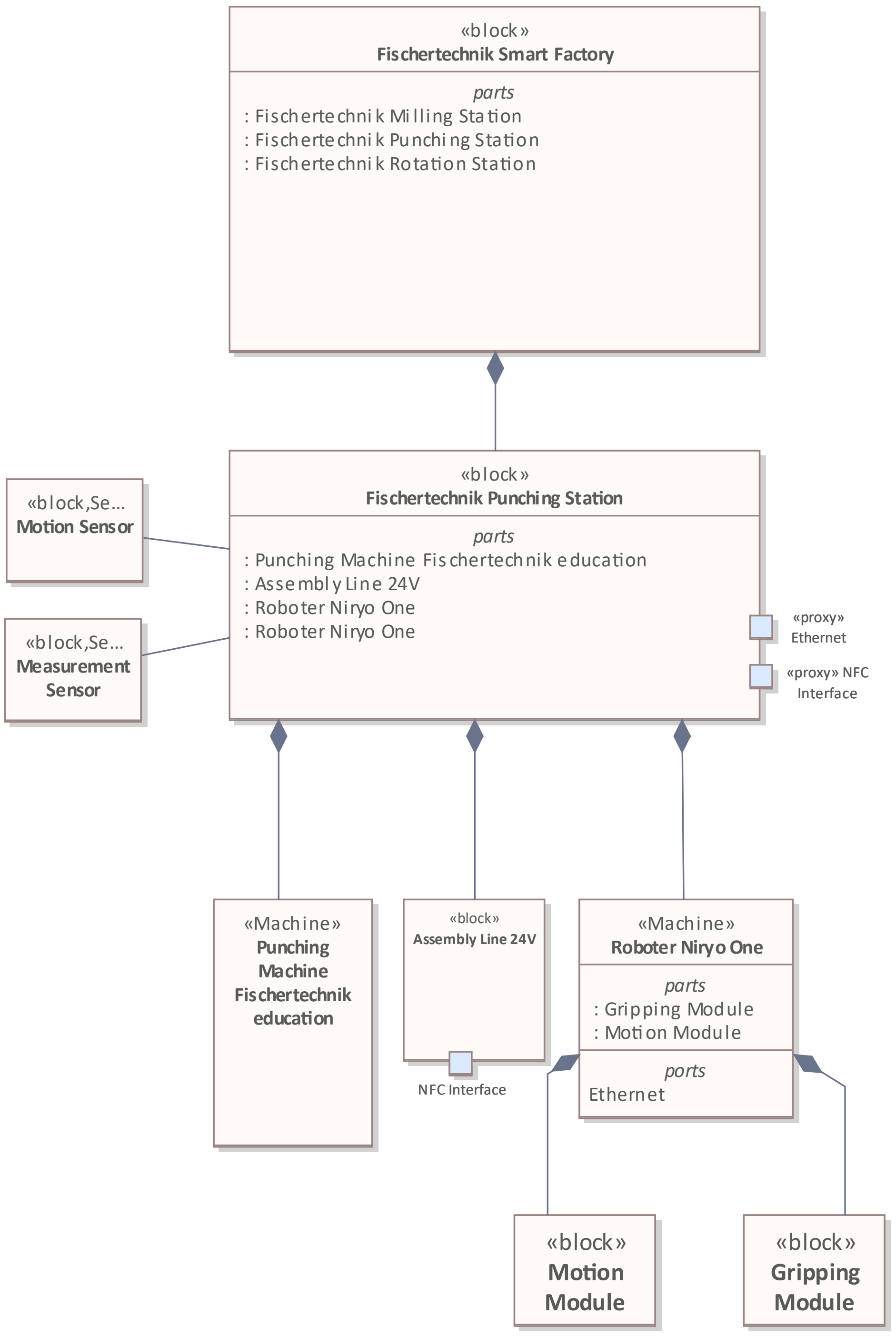
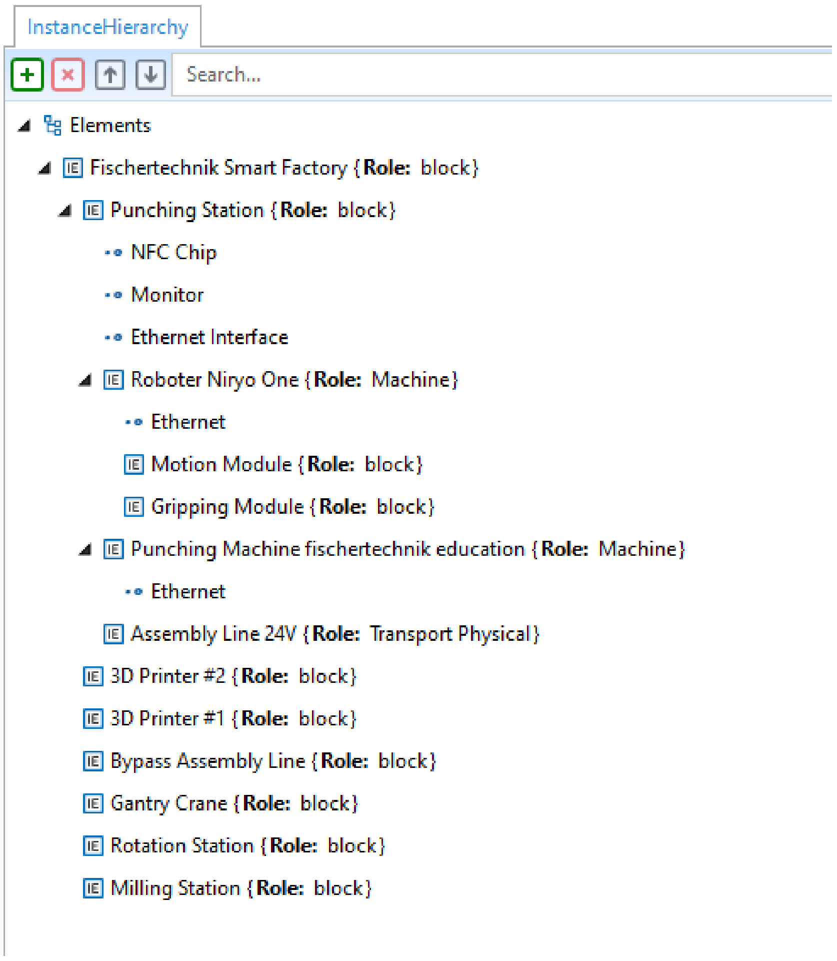


Figure 8.26: SysML block definition diagram of the Fischertechnik smart factory

A unique feature is the export of stereotypes from system elements and embedding them as roles within the instantiated elements of the InstanceHierarchy. This enables interconnection between those elements to the AutomationML role classes. Those roles could be imported from other AutomationML-files or newly created within the SysML model. For example, the robot contains the role “Machine”, while the Assembly Line is of the Role “Transport”. However, if no stereotype is defined, the base role is exported, which is a SysML block in this case.



- Elements
  - Fischertechnik Smart Factory {**Role:** block}
    - Punching Station {**Role:** block}
      - NFC Chip
      - Monitor
      - Ethernet Interface
      - Roboter Niryo One {**Role:** Machine}
        - Ethernet
        - Motion Module {**Role:** block}
        - Gripping Module {**Role:** block}
      - Punching Machine fischertechnik education {**Role:** Machine}
        - Ethernet
      - Assembly Line 24V {**Role:** Transport Physical}
    - 3D Printer #2 {**Role:** block}
    - 3D Printer #1 {**Role:** block}
    - Bypass Assembly Line {**Role:** block}
    - Gantry Crane {**Role:** block}
    - Rotation Station {**Role:** block}
    - Milling Station {**Role:** block}

Figure 8.27: Fischertechnik industrial plant within AutomationML

The second scenario involves importing an externally developed system model into RAMI 4.0-layout architectures. If this model does not yet exist and is available in the AutomationML structure, this option can be chosen, as it works for each model accessible in AutomationML. The SysML block definition diagram is automatically created from the AutomationML-file, where it can be further processed within the MBSE development process. Either further refinements could be done on other granularity levels, or the components could be traced to the different layers of RAMI 4.0 to create a comprehensive, flexible production system description according to this SoA. However, the third scenario summarizes both previous scenarios and synchronizes both system models if they have been edited in each tool, the RAMI Toolbox or AutomationML.

No external tools have been used to process the system further to keep the definitive case study superficial. All changes to the system model in AutomationML have been made with the corresponding AutomationML Editor, while the RAMI 4.0-based architecture has been edited with the RAMI Toolbox. It can be claimed that the described application successfully validates the functionality of the RTE approach utilized within the RAMI Toolbox.

8.4 Findings

The chapters described so far outline the development of the RAMI 4.0 modeling framework and its application. A wide variety of aspects were considered to address stakeholders and their concerns. Based on a wide variety of established standards, the resulting framework provides a comprehensive concept to create flexible production systems based on RAMI 4.0. The research question specified that the modeling framework must support domain-specific peculiarities and enable holistic and interdisciplinary engineering of the flexible production system. The extent to which the results of this thesis can cope with these aspects is evaluated in the following paragraphs. Thereby, the well-known method for analyzing implemented software architectures, better known under the term SAAM, is used. As this methodology specifies, different application scenarios of the architecture are considered, and it is validated whether the proposed architecture might achieve its purposes. In the context of this work, typical Industry 4.0-based scenarios are chosen since the purpose of RAMI 4.0 is the creation of contemporary and future industrial systems. In the following, therefore, each of the respective scenarios is discussed in detail, whose PoC application has been outlined in the previous sections of this chapter, and their degree of fulfillment about selected quality attributes is delineated. The quality attributes of choice to evaluate the modeling framework in the context of this thesis are feasibility, usability, dependability, traceability, and composability. The feasibility indicates if the resulting framework can implement the respective scenario, while the usability demonstrates the degree to which users are supported. Dependability and traceability refer to the instantiated production systems based on the framework and aim to exhibit how essential characteristics are ensured. Finally, the composability gives an idea of how the framework concepts fulfill the stakeholders and their concerns. By executing this evaluation strategy, different aspects of the RAMI 4.0 are considered. An overall statement can be given to what extent the proposed solution can serve and support other application possibilities.

The substantiation for using the exact mentioned quality attributes is delineated in the following, with detailed explanations.

Feasibility

Feasibility is an important aspect to consider in the context of Industry 4.0, as the primary new outcome is generated, which needs to be evaluated towards appropriation for application. Thus, different aspects related to this trend need to be validated about feasibility, which counts for new technologies or methodologies. Therefore, it is suggested to consider this particular system quality attribute to allow the integration of automation and control processes for the continuous improvement of decision-making (Saucedo Martıénez and Noriega, 2020).

Additionally, Bowen et al. (2009) explain that feasibility studies determine whether a research project is appropriate for further investigations or testing. In more detail, researchers can assess whether or not the ideas and findings of such studies might be shaped to be relevant and sustainable. There are several reasons to perform such a feasibility analysis. For example, such investigation is used when community partnerships need to be established, increased, or sustained, or few published studies use a specific intervention technique on detailed data. The latter also counts when such studies have not been performed with the suitable depth or needed level of detail. However, if previous studies have not been successful or the outcome stated that further research is required to achieve evaluable results, investigating the feasibility of the object under investigation is also a suitable parameter to examine. The authors also explain that choosing an appropriate area of focus must be evaluated. As there are eight different areas of focus to select in general, the chosen area should be implemented within the context of this thesis.

When selecting implementation as an area of focus, the evaluation considers the extent, likelihood, and manner of how and to what extent an intervention can be fully implemented as planned or proposed, often done in an uncontrolled design. To evaluate whether the implementation's feasibility will work, a small-scale demonstration project is utilized to examine if the implemented artifact might be deployed in any clinical or community context (Bowen et al., 2009). In this thesis, the implemented artifact is the proposed RAMI 4.0 modeling framework, while the demonstration project could be considered the utilized case study as an application scenario.

Usability

Usability is one of the more critical aspects to address when engineering systems, either when dealing with manufacturing systems (L. Ramos et al., 2020) or when handling SoS (Bianchi et al., 2015). It is claimed that it is essential to use applications that hold complete information and allow the user access to specific production data without much navigation, which is especially difficult within a SoS environment. Additionally, usability is addressed within the ISO/IEC 25010 (Haoues et al., 2017).

The term usability has been introduced to replace the meaning of user-friendliness. Three different views define it: the product-oriented view, the user-oriented view, and the user performance view (Bevan et al., 1991). At first, the product-oriented view indicates that usability might be measured regarding the ergonomic attributes of the product. Next, the user-oriented view measures usability in terms of the mental effort and attitude of the user. Finally, the user performance view tries to explain the term usability by investigating how easy to product is to use or whether the product will be used in the real world. Ease-of-use thus determines the degree to which users can use the system with the skills, knowledge, stereotypes, and experience they can bring to bear (Bevan et al., 1991). The measurement of usability is thereby a difficult task, as it is hard to specify adequate criteria in advance. However, in the most general case, the criteria will depend on the user's specific requirements, task, and environment of use. In the context of this thesis, the usability of the modeling framework mainly relies on the evaluation criteria of whether it will be used in real life, which also examines the acceptability as part of the user performance view and the user-oriented view and the needed effort to master the framework.

Dependability

Regarding the infrastructure of Industry 4.0, this is one of the essential criteria for systems evaluation (L. Ramos et al., 2020). Infrastructure guarantees the correct execution of the system in different layers of the organization. To disassemble this sentence, the term "correct execution" refers to dependability. In contrast, the terms "different layers of the organization" and "guarantees" are fulfilled by providing high composability and ensuring traceability. Therefore, these quality attributes must be considered to evaluate flexible manufacturing systems. In addition, they are also mentioned for investigating the quality of SoS, as current quality models, like the ISO/IEC 25010, are not able to address their complex interdependencies (Bianchi et al., 2015).

Dependability could be considered an umbrella term for system properties like reliability, availability, maintainability, survivability, safety, and security. Combining those attributes and summarizing them leads to the measurement of dependability. Thereby, system characteristics like threats, faults, errors, and failures or other system attributes are examined to achieve adequate heights. Additionally, measures to counteract these characteristics, like fault prevention, fault tolerance, fault removal, or fault forecasting, are also considered when dealing with this term. Dependability could thereby be measured either qualitatively or quantitatively. A qualitative evaluation aims to identify and classify those characteristics and attributes that lead to system failures. Nevertheless, the quantitative evaluation seeks to calculate the probabilities to which some dependability attributes are satisfied, subsequently considered dependability measures. In this particular thesis, the dependability is measured by investigating the respective application scenarios of SAAM and the faults that emerged when modeling each scenario according to the given case study. (Avizienis et al., 2001)

Traceability

Aizenbud-Reshef et al. (2006) states that traceability relationships help stakeholders understand many associations and dependencies among system artifacts. Thereby, the extent of traceability practice is viewed as system quality measurement, which several standards could mandate. Model-driven development provides novel possibilities to establish and use traceability information within a system model. In more detail, this quality attribute is mainly achieved by defining and maintaining relationships between the artifacts involved within the system‘s life-cycle during all development phases. To ensure the traceability of a system before actually modeling its architecture, securing this information within the metamodel is considered a promising task. Thereby different types of relationships, with or without semantics, are provided to link the respective artifacts. However, in the context of this thesis, the measurement of this attribute is achieved by the degree to which the artifacts are interconnected within the case study scenario. A flexible production system consists of many elements, including product attributes and manufacturing system features, so maintaining traceability might be important.

Composability

At last, composability defines the degree to which the components of a heterogeneous complex system can express a system's structure and behavior. The system models' composition is considered essential to execute this step according to Sarjoughian (2006). According to the authors, it is mentioned that manufacturing supply chains are well-known network systems that are often modeled at varying levels of detail and from different aspects. Thereby, the composability regards the prominent modeling paradigms *separation of concerns* as well as *divide and conquer*. Thus, this thesis makes us of this concept by analyzing the degree of various system models that might be applied for varying application scenarios.

In the context of this thesis, the respective quality attributes are not weighed equally, as various independent aspects have been considered. Unifying and adjusting the weighting to be equal would therefore distort the result. This means that the quality attributes are individually assessed, and their objective importance for their particular within this thesis is estimated. The resulting weighting is thus outlined in Table 8.1. The table shows each of the five quality attributes and the selected weight in percentage. As the research question and the primary reason for performing this work was to evaluate whether it is possible to create an architectural framework that allows systems engineering of flexible production systems, feasibility is the essential attribute with 45% of a total of 100%. This value has been selected to emphasize the importance of investigating a possible utilization of RAMI 4.0 and providing read-to-use concepts. Next, the second-highest weighted attribute is represented by usability. This means that the resulting modeling framework should be used for any interested practitioner and support the complex task of systems engineering. This is why a value of 22.5% has been linked to this quality attribute. Subsequently, traceability exhibits a third of the importance of feasibility, which places it in the middle of all attributes. Traceability is essential to interconnect the key concepts of the RAMI 4.0 modeling framework and support complex system representations of flexible production systems. However, as parameters like fault tolerance or security are not within the main focus at this early stage of the framework development, dependability has been assigned with a lesser weighting. Finally, composability is also considered to distinguish the various elements of the modeling framework. Due to the minor importance, only a value of 7.5% has been selected.

After defining the conditions of the evaluation strategy and applied methodologies, the single evaluation scenarios are validated according to these requirements. By doing so, all three application scenarios of the modeling framework are independently

Table 8.1: Weighting of the system quality attributes

Quality Attribute	Weighting in %
Feasibility	45%
Usability	22.5%
Dependability	10%
Traceability	15%
Composability	7.5%

evaluated, and the degree of fulfillment of each of the quality attributes is determined. Thereby, four different values for the degree of fulfillment are assigned:

- Insufficient: The quality attribute is not or partly fulfilled and not ready for industrial utilization.
- Mediocre: The quality attribute is fulfilled but needs to be enhanced to be practically applied.
- Sufficient: The quality attribute is sufficiently fulfilled but could be optimized.
- Mature: The quality attribute is ready for industrial utilization.

Finally, to give overall feedback, the combination of each of the scenarios is assessed by analyzing each other's interactions. The evaluation is additionally underpinned by providing a more detailed delineation of identified aspects. Thus, the following gives a detailed description of each evaluation outcome about the respective prototypical implementation.

Evaluation result about system quality attributes

The metal profiles for the subway tracks represent the first application scenario to evaluate the outcome of this work. This use case aims to apply the domain-specific concepts of the modeling framework and validates them toward the quality attributes. The result is thereby shown in Table 8.2. This table indicates the degree of fulfillment for each respective quality attribute in the context of all scenarios, the metal profiles for subway tracks to evaluate the DSL concepts; the Siemens Fischertechnik use case to validate modeling tasks, and the toolchain integration of the RAMI Toolbox. Finally, the individual results are accumulated, and the system quality attributes are evaluated.

The main focus for executing this validation was to validate three significant con-

Table 8.2: Scenario-based evaluation of the system quality attributes

Quality Attribute	DSL concepts	Modeling	Toolchain	Overall
Feasibility	sufficient	sufficient	mediocre	sufficient
Usability	mediocre	mature	mediocre	sufficient
Dependability	insufficient	mediocre	insufficient	insufficient
Traceability	mature	mature	sufficient	mature
Composability	mature	mature	sufficient	mature

cepts of the modeling framework. At first, the metamodel, including elements for modeling flexible production systems, the architecture refinement for each of the RAMI 4.0 layers, and the process model. The second application scenario, the prototypical implementation of the Siemens Fischertechnik model, has been chosen to evaluate the implementation of the modeling framework itself. Integrating architectural standards and methodologies, like SPES or the Zachman Framework, has been validated along with adopting the ISO 42010 architectural aspects to the modeling framework. At last, the third application scenario chosen in the context of SAAM is represented by several case studies, which mainly evaluate the modeling framework towards its integration within an engineering toolchain and its interoperability with other tools.

Feasibility

However, the outcome of this evaluation by applying the metal profiles for subway tracks exhibited mixed results. In detail, the implemented domain-specific concepts demonstrated sufficient feasibility. This value falls back on the accumulated modeling elements, which provide valuable notations for various applications of industrial systems, in particular for creating their architectures. By refining the theoretical concepts of RAMI 4.0 and providing a detailed architecture definition of each layer, those architectures could be structured in detail and fulfill Requirement 4.5.1. The application of the case study has shown that those architectural definitions provide a suitable environment for addressing all concepts of a flexible production system, which also contributes to the feasibility.

The result of the RAMI Toolbox evaluation proposed sufficient feasibility, mainly reaching for providing a ready-to-use framework to any interested systems engineer. By integrating well-known or established standards and methodologies, the acceptance is improved, and the consideration of a wide variety of stakeholders results in providing concepts for many application scenarios.

As far as the toolchain integration is concerned, several concepts are still at the starting point or do not allow a comprehensive application of all needed scenarios. For example, OPC UA export is limited to one particular modeling element, co-simulation is solely available when filling specific attributes, or the AutomationML export interface is currently only available as PoC and not suitable for utilization in existing production systems. Therefore, the feasibility could only be fulfilled with half of its possibilities. Further developments and other application scenarios need to be considered to reach a complete degree of fulfillment.

Traceability & Composability

In addition, the guided modeling steps of the process model ensured an interconnection between all instantiated system components, which resulted in assigning a mature degree of fulfillment in traceability. The modular development approach and the structurization according to the paradigms *separation of concerns* and *divide and conquer* strongly contributed to the composability of the modeling framework and concurrently to the fulfillment of Requirement 4.5.5, where the system's structure might be expressed by components that are utilized for various aspects within its architecture.

Dependability

However, fault tolerances and safety and security have barely been considered at this early stage and during the provision of domain-specific aspects, as the main focus was primarily to ensure feasibility and usability. Thus, the dependability has been selected to be insufficient, which should be enhanced in the prospect of this thesis. As far as the toolchain integration is concerned, the provided methodologies are currently not prone to errors, while safety or security is partly addressed.

Usability

Finally, as the modeling elements and the process model are available to practitioners, the usability of the domain-specific aspects is partly ensured, contributing to Requirement 4.5.4. Although, the usability should be improved by allowing an easier revision of the metamodel and adding new modeling elements or process steps within each of the

RAMI 4.0 layers. Thus, adding up the remaining percentages to reach maturity should be done in follow-up projects by utilizing more sophisticated application scenarios.

As the modeling framework is made applicable with the help of the RAMI Toolbox, practitioners can utilize most of the integrated concepts, like guiding through the development processes by providing a step-by-step process model, providing a large number of models to capture all domain-specific aspects of flexible production systems or automatically aligning system architectures towards the theoretical concepts.

However, similar statements could be made in the context of the toolchain integration, as the RAMI Toolbox provides an interface automating the export functionalities or interconnections to other tools. At the current stage of development, those functions have been developed at the beta level and need to be polished for actual industrial usage. All in all, it could be said that the toolchain integration of the RAMI 4.0 modeling framework is the least established aspect and thus resulted in lower degrees of fulfillment.

By investigating the overall column, it is exhibited that composability is the most mature quality attribute concerning the RAMI 4.0 modeling framework. An explanation for this could be integrating methods that provide structural organization or the refinement of RAMI 4.0 to compose systems onto multiple hierarchy levels, contributing to Requirement 4.52 and Requirement 4.5.3. The composability is followed by traceability, defined by Requirement 4.5.7, which is ensured by integrating ISO 42010 or providing a metamodel, representing immutable connections between several modeling elements. Next, the usability is sufficiently fulfilled. The RAMI Toolbox mainly achieves this value, which deals with applying the developed concepts and ensures user interactions, ensuring Requirement 4.5.6. However, there is further potential to enhance dependability. As previously mentioned, since safety, security, and fault tolerance are barely considered during the development of the modeling framework, this quality attribute is insufficiently fulfilled. It needs to be stronger investigated in future development cycles. However, evaluating the feasibility of the RAMI 4.0, the modeling framework resulted in a sufficient degree of fulfillment. Most implemented aspects appear promising and are fostered in different application scenarios.

Evaluation Results & Discussion of Research Question

The applied case studies and the evaluated modeling framework allow an adequate interpretation of the results and an answer to the research questions. While the prototypical implementation of the three varying application scenarios is outlined within this chapter, an examination of the individual results could be performed. By applying the evaluation strategy, the application of SAAM deals with an overall evaluation of the RAMI 4.0 modeling framework by partly validating its composed parts, its domain-specific concepts, the implementation according to established tool sets, and the toolchain integration into the engineering workflow. However, the superficial nature of the case studies and the evaluation team exhibited limitations for this strategy. This means that even though it has been tried to achieve comprehensive anticipation of the complete process for creating the modeling framework, without an actual industrial development team, its suitability for application could solely be estimated. Nevertheless, the application of the case studies gained valuable insights for evaluating this suitability, which counteracts the missing industrial examples. Thus, in contrast to the previous empirical evaluation of the system quality attributes, the following paragraphs discuss the observational evaluation. As those observations originate from heterogeneous sources, neither a weighting and comparison nor an application of a qualification schema could be made to validate the results. Hence, these observational steps intend to record experiences that emerged during the utilization of the modeling framework and its application in the sense of the application scenarios, which allow a critical reflection and derive the prospect of this work.

- One of the most significant issues concerning RAMI 4.0 is the missing formalization of the reference architecture itself. In the official standard describing the model, multiple definitions are not underpinned by practical applications. Therefore, applying these concepts is difficult, as they look good on paper but are too generic to allow a concrete instantiation. This issue is mainly counteracted by the layer refinements and the integration of established standards, which is part of this thesis. However, the implementation and evaluation of those layer refinements exhibited interesting results. In more detail, the three dimensions of RAMI 4.0 inherent inconsistencies. This falls back to the problem that different topics are addressed when dealing with flexible production systems. Such a system consists of smart products to be developed, production processes that are executed by the system and deal with creating the products, and production systems that deal with developing those products by executing the processes. Thereby, RAMI 4.0 combines all three topics within one architecture model with all addressed aspects. Nevertheless, this combination of the flexible production system parts resulted in the mentioned issue. Those inconsistencies emerged by applying the various case studies according to the proposed modeling framework. The application scenarios demonstrate that their respective system parts must be developed according to particular structures but not on all three axes of RAMI 4.0. In more detail, a smart product could be expanded throughout its life-cycle by addressing all different states, from planning to removing it. Although, by doing so, the product itself could only be addressed at the product column of the automation pyramid axis of RAMI 4.0. As such a product does not span the whole enterprise or the work unit level, no abstraction layers are needed to model it. On the other hand, hierarchy levels are implicitly needed to structure and compose the complex system required for developing the entire production system. Nevertheless, considering the life-cycle of such a system would result in ample confusion and an unnecessary complication of the system architecture, as the surrounding process of the machines might not be considered during model-driven engineering; only their instances are needed that provide the required functionality. Concluding, the inconstancy of RAMI 4.0 could be counteracted by splitting the reference architecture and modeling different aspects on different axes. A smart product should be modeled at the product axis, including its life-cycle. In contrast, the production system should be modeled on the remaining hierarchy levels of the automation pyramid without the product column and the system life-cycle. The interoperability layers could be used for both smart products and production systems. The production systems, thereby, could deal as an interface connecting the product with the system, as it might be derived from the product architecture and could be embedded within the system architecture representing the system's intended functionality on the Business or Function Layer. The product architecture describes the process step-by-step, indicating the tasks needed to create the product from the raw materials. This process might be transformed to use case scenarios or functional architectures embedded within the system architecture. This should be investigated in future projects with more sophisticated application scenarios.

- During the modeling activities, it became apparent that some tasks are repetitive, including a significant manual effort. Examples are the elaboration and descriptions of functions and similarities by defining the Information or Communication Layer or integrating a typical set of requirements to fulfill a particular KPI. To support future users, those repetitive tasks could be enabled by additional functionalities provided by the RAMI Toolbox or integrate model patterns within the DSL. This will allow easy instantiation of a specified set of elements or fulfill a particular automated functionality at a single click.
- The RAMI 4.0 modeling framework provides different utilization scenarios, either for model-driven engineering of greenfield systems or for digital twin development of brownfield systems. However, architectural modeling of the prototypical

use case implementation has shown that flexible production systems are usually developed using both methods. This falls back to the fact that usually, there are already production lines deployed within current manufacturers for designing their products. To meet industrial requirements, those traditional production lines need to be refined. However, this refinement might not be achieved by only applying one of the respective scenarios; instead, a combination needs to be used. This also is substantiated by the fact that already existing production lines might be considered brownfield systems. In contrast, newly needed system parts could be engineered by utilizing model-driven engineering of a greenfield. Thereby, there is a constant interplay of both methods since the digital twin of the system needs to be developed before implementing new system parts. This means future users might be guided through both scenarios and should be aware that a dynamic utilization within the panes of the modeling framework is needed to foster its application. This means providing a more flexible development process, and entry into architectural modeling within any of the introduced panes will be necessary.

- After developing a system based on the RAMI 4.0 modeling framework, the system architecture should be available to other tools within the toolchain to profit from previously developing the system's architecture. Thus, the modeling framework provides several exporting capabilities that use various modeling elements. However, the created bi-directional interfaces mostly rely on those modeling elements being fully specified and available in the correct form. As the modeling environment is difficult to be applied for non-trained practitioners, there need to be model checkers available that ensure the proper application of the modeling framework. The applied scenarios have shown that it is also difficult to arrange the modeling elements adequately for their export; such a functionality is needed. This means a future version of the modeling framework needs to contain a particular model-checking tool.
- Moreover, as various system stakeholders need to be provided with information from the system architecture, like attributes or element positioning, the interfaces should consider those values to be exported. This is also needed because the modeling environment is hard to access for stakeholders with bare experience with such tools, like managers or consumers; those interfaces should be adjusted adequately. While interfaces already deal with exporting OPC UA information or whole plant compositions into AutomationML, many other possibilities are available for exporting data from the model. For example, technology operators might be provided with information standards, while requirement engineers could be supplied with user stories to calculate analytic hierarchy process (AHP)-matrices. In conclusion, it is needed further to enhance the RAMI 4.0 modeling framework

and find other scenarios where interfaces from the model are required to address the concerns of a wide variety of stakeholders. This step needs to be investigated in the prospect of this thesis.

- The current version of the modeling framework allows the instantiation of particular system components of a flexible production system. After that, all system architectures might be modeled from various domains. Such domains might be automotive, steel industry, or individual shoe development. Consequently, the modeling framework is suitable for developing systems at a particular level of abstraction. In the case of the prototypical application scenarios, two different possibilities have been shown. On the one hand, some modeling elements have been unavailable for specifying detailed system architectures and are missing within the current DSL. On the other hand, too generic elements were identified, not inheriting the needed attributes for describing the system at the current level. This means, in the future new possibilities for either instantiating new modeling elements or abstracting new elements from already existing ones. As those steps are difficult to be achieved from a developer's point of view, allowing actual users to create their modeling elements could be a significant advantage to counteract this issue. To achieve this, two major implementations should be planned for follow-up projects enhancing the framework. At first, automatic metamodel manipulations that allow any practitioner to generate their modeling elements should be enabled by an improved version. Subsequently, particular reference architectures providing an element set and correlations for a specific domain need to be provided, which supports the development of system architectures within focused application areas.

Considering the research question of this thesis, the evaluation enabled a meaningful assessment to answer it. With the RAMI 4.0 modeling framework, a suitable tool to develop current and future manufacturing systems has been provided. In addition, the RAMI Toolbox enables the application of this framework and its utilization for any interested user. The utilization within multiple application scenarios has shown that the implemented concepts appear suitable for creating architectural models of such flexible production systems. While the currently used concepts still exhibit some limitations, future enhancements of the modeling framework are suggested. Its parts are subdivided and individually addressed to explain how the research question is answered.

1. The metamodel considers domain-specific peculiarities for describing flexible production systems. Within this description of the modeling elements, particular attributes and utilization possibilities for every single element are provided, which

enables a comprehensive element set for describing such a system. Additionally, within the refinement of each of the RAMI 4.0 interoperability layers, domain-specific characteristics to address the concerns of such systems are integrated. This means before actually implementing the modeling framework, an extensive investigation of the factors originating from Industry 4.0 has been performed, which resulted in the specification of modeling elements that directly target the engineering of current or future manufacturing systems.

2. Those domain-specific peculiarities have been consolidated with the model-based engineering concepts of various established methodologies. Analyzing the Zachman Framework, TOGAF or SPES resulted in providing promising or less-suitable methods to achieve this. However, by utilizing MDA, MBSE is enabled across the axes of RAMI 4.0. In more detail, various development strategies are available to either design a system from scratch or model architectures of already existing systems. Combining the guided process model and the suitable domain-specific element builds the base for introducing the comprehensive modeling approach.
3. Hence, the comprehensive modeling approach is ensured by implementing the modeling framework within the RAMI Toolbox. The developed concepts are applicable and provided to any interested users for utilization. Regarding the application scenarios, the proposed approach can model the architecture of various flexible production systems and provide a suitable notation for describing different aspects of those systems. Thus, it could be stated that the modeling framework can be applied to engineering various industrial systems and represents such a comprehensive modeling approach.
4. The part describing the holistic and interdisciplinary engineering is addressed by providing various abstraction levels as well as interoperability layers, which compose the system into more granular partitions on the one hand and ensure traceability on the other hand. This traceability is also ensured beyond the modeling framework and other tools within the toolchain, where multiple interfaces allow model transformation and the external use of engineered results. An example of such an export functionality is the implementation of an AutomationML-interface, which enables the shift from basic engineering to the interdisciplinary disciplines of detailed engineering. Thus, the modeling framework assures holistic ness and interdisciplinarity throughout the entire toolchain.
5. Finally, to address all stakeholder concerns, an extensive analysis of all architectural stakeholders and their concerns have been performed. As a result, practical viewpoints and model kinds have been derived that address those stakeholder concerns. An established standard underpins this step by falling back on the the-

oretical concepts of ISO 42010. This means that each pane on each abstraction level provides suitable models that regard the respective stakeholder and its interest in the architecture.

Even if the concepts presented in this work are promising, it becomes clear that further research is still needed to comprehensively answer all parts of the research questions and reach a higher degree of fulfillment. More sophisticated case studies need to provide new requirements to develop novel concepts. In addition, the remaining aspects in this work must be further investigated in the subsequent iterations of ADSRM, which will be done in the prospect of this thesis.

Chapter 9

Summary & Outlook

This thesis deals with developing and proposing a modeling framework that enables model-driven engineering of flexible production systems based on RAMI 4.0. In the context of this work, the research question "How can domain-specific particularities and model-based engineering concepts be consolidated to a comprehensive modeling approach enabling holistic and interdisciplinary engineering of current and future manufacturing systems by addressing all stakeholder concerns?" is tried to be answered. Therefore, this chapter aims to summarize the performed tasks to answer this research question by splitting it into different parts and how the RAMI Toolbox can deal with the issues. To do so, the following sections summarize the chapters and give insights into how the particular research aspects of the question are dealt with. Special focus is set on the related work of this thesis and how the presented and accomplished research work of this thesis contributes to the extension of the related work. As Hevner and Chatterjee (2010) specifies, scientific contributions should add content to an already existing knowledge base. To do so, a list of publications is introduced in Chapter 9.1, which lists all contributions to this knowledge base on which this thesis is founded. However, the proposed work adds additional value to this list and thus is shortly summarized within the following paragraphs.

Chapter 1 outlines the problem context of this thesis and gives an overview as well as limitations of current research projects. The targets and objects of the proposed research work are derived, outlining the scope for fulfilling this work and describing the motivation. Additionally, it states what is tried to achieve and what remains out of scope regarding the research focus. This chapter's main contribution is the definition of the problem statement and the subsequent derivation of a research question.

After that, Chapter 2 refers to the state-of-the-art when performing systems engineering within the industrial area, particularly in the context of Industry 4.0. While systems engineering is generally outlined, MBSE focuses on this thesis section. Thereby domain-specific contexts with relation to Industry 4.0 are elaborated and compared with established methods. By doing so, unique systems engineering disciplines, like RTE or requirements engineering, are additionally quoted. Next, best practices for describing and developing reference architectures, like MDA, metamodeling, or DSLs, are described in detail. This section follows an overview of existing standards that might be applied and represent best practices in architecture development. Examples of those standards are the IEEE 1471, the ISO 42010, or the FAS methodology. At the end of this section, methods to apply or evaluate created systems are delineated, which deal with investigating architectures of production systems before their actual implementation. Those methodologies include definitions for the system life-cycle of such a system and the derivation of an engineering toolchain, including several tools such as simulation applications. To round up this chapter, a short introduction to the term Industry 4.0 is given at its start, explaining this trend and the accompanying transformation in detail. Hence, by comparing the proposed approaches, current shortcomings could be identified, and actual research projects could be validated for their feasibility. This leads to the definition of the scope of the present research work.

Subsequently, Chapter 3 gives an overview of current concepts applied for modeling production system architectures. This includes references to meaningful literature and established ideas for defining such architectures. At first, the main foundation of this research work, RAMI 4.0 and SoAs in general, are delineated in detail. This is followed by current approaches to describe flexible production systems, as exemplified by the PPR systems approach, as well as the distinction between basic engineering and detailed engineering. Finally, currently introduced methods for structuring production systems are delineated, which include the well-known frameworks SPES, the Zachman Framework, or AutomationML. This chapter mainly contributes to aligning different concepts for developing system architectures. It contextualizes these concepts in the industrial domain and aligns them with the proposed research problem.

Chapter 4 falls back to the previous chapter and combines the problem context with the state-of-the-art and current approaches to define the research methodology for achieving valuable results. The overall approach for executing the research methodology is outlined shortly, and the concepts used for fulfilling the single steps are provided. Specifically, those concepts contain DSR for falling back on general research paradigms and ADSRM for actually performing research tasks. To evaluate the results, SAAM is applied, and three different prototypical implementations are validated against good system quality attributes. The case studies, including metal profiles for subway tracks, the Siemens Fischertechnik model, and a packaging process use case, are applied to gain relevant results. Thus the main contribution of this chapter is the definition of a research methodology and substantiates the utilization of design science.

The first chapter deals with the implementation of the modeling framework, Chapter 5, firstly outlines developed results. Those results inherit the refinement of RAMI 4.0 and its interoperability layers, which are barely defined and only provided from a theoretical perspective. Thus, a particular metamodel of RAMI 4.0 is developed, which is made applicable via a specific UML profile. However, the refinement of the layers refers to a more detailed architecture definition, including viewpoints and model kinds for describing the domain-specific characteristics of flexible production systems. At last, this chapter provides a particular development process that guides users through the various modeling steps when elaborating such systems. This chapter thus contributes to the future implementation of the modeling framework by providing aspects needed when addressing complex systems in the manufacturing domain. By developing a particular DSL, UML profiles or a process model inheriting both the ISO 15288 as well as TOGAF, a standardized base is set.

In Chapter 6, this base is apprehended and enhanced with additional details to implement the RAMI 4.0 modeling framework. Thereby, while different integration of already established methods has been more or less successfully achieved, the development of a particular structure for dividing flexible production systems is the centerpiece of this section. A proprietary modeling framework is constructed to build on the integration outcome of the Zachman Framework and SPES, which both inherit shortcomings. The framework structure uses the abstraction levels and is extended by the architectural concepts of ISO 42010. This means that stakeholders interested in the architecture of Industry 4.0 systems are elaborated, and viewpoints or model kinds to address those concerns are provided. By doing so, the previously offered domain-specific aspects provide those stakeholders with a suitable modeling environment. In addition, different development strategies, like model-driven engineering, digital twin development, and system decomposition or factory integration, are provided to guide the modeling process. In summary, all mentioned concepts of the previous two paragraphs, which are integrated within the modeling framework, are made applicable to interested practitioners with the implementation of the RAMI Toolbox. This software aims to support users in any needed steps for developing flexible production systems. Hence, the contribution of this chapter is the actual implementation of the modeling framework and the provision of the RAMI Toolbox, including a structured development approach, domain-specific elements, and guided development processes.

Next, the application of the framework and its integration within the engineering toolchain is explained in detail in Chapter 7. Typical application possibilities, like simulating a modeled system or controlling a PLC based on modeled tasks, are examined. To achieve this, interfaces must be implemented and applied to address the other tools within the toolchain. This also counts for different possibilities of the modeling framework, like exchanging engineering information with AutomationML. Thereby, scenarios for using AutomationML to perform a model transformation within the basic engineering of flexible production systems or exporting basic engineering information to be used within detailed engineering are investigated. Nevertheless, the possibility to describe systems according to the well-known PPR notation is additionally ensured within this chapter. As providing interfaces to a large number of engineering tools would exceed the scope of this thesis, the selected application possibilities represent meaningful examples. The contribution could be stated with the possibility of the modeling framework interacting with other tools and providing suitable interfaces.

Within the penultimate chapter, Chapter 8, the evaluation of the implemented concepts is achieved. Thereby, the evaluation method SAAM is utilized, which uses various application scenarios to validate the modeling framework's utilization for different examples. Based on the case studies, requirements have been derived, and system quality attributes have been measured. Three different application scenarios are considered, all validating other modeling framework concepts. The subway track example thus evaluates the utilization of the domain-specific concepts, which is also done with the Siemens Fischertechnik model. However, this case study is mainly applied to validate the implemented modeling framework and the RAMI Toolbox. In contrast, the packaging use case is used to analyze the toolchain integration of the modeling framework. Subsequently, the evaluation results are compared and gained findings of the evaluation method are outlined in detail, representing the chapter's contribution.

Finally, Chapter 9 gives an overview of the entire thesis and summarizes each chapter by providing insights into implementation details and contributions of the respective subdivisions. To also address the prospect of this work, the possibility for future projects, and the shortcomings of this thesis are addressed.

9.1 Prospect

As the work presented in this thesis provides a promising approach for developing current and future manufacturing systems, an innovative way for industrial systems engineering is generated. The proposed modeling framework allows it to be applied in various application scenarios and inherits suitable concepts for addressing all aspects of flexible production systems. However, although providing a ready-to-use methodology to be used by non-export practitioners, further enhancements could be achieved. This also is substantiated by the inherent limitations given by the research framework, which results from the vast field of Industry 4.0. Therefore, only selected aspects could be considered while performing this thesis research methodology. Hence, future improvements are promoted and should be implemented whenever possible to generate a broader vision and a more general or specific application of the modeling framework. To fall back on the claim of Linkin Park (2010), "the hardest part of ending is starting again," different considerations have been identified as suitable for future enhancements.

The current modeling framework current version mainly targets the production systems themselves without focusing on smart products or production processes. This hinders the dynamic interchangeability between the respective system parts and thus counteracts the concept of flexible production systems. A future version of the modeling framework should strongly consider this interconnection and provide suitable notations to interdisciplinary engineers in such systems to achieve simultaneous development of the system, the process, and the product.

Additionally, established frameworks and well-known tools for describing and developing industrial systems must be investigated to contribute to a more comprehensive toolchain. Interfaces to those external tools should be found, which allow the import or export of engineering information throughout the toolchain. Additionally, as some stakeholders might not be able to utilize the modeling framework due to missing knowledge or no access to the modeling environment, an interface to an open-source tool could be integrated.

More research needs also be put into the PPR topic. As several domain stakeholders or system engineers use this notation to describe flexible production systems, a better link could be created. While the currently introduced methodology only allows modeling the production system itself and interconnecting it with the product or process model, PPR could combine all three concepts within a single model. A more detailed engineering process could be represented by doing so, enabling new possibilities. One example would be the interconnection of simulators for a co-simulation, which are resources within the PPR, and their behavior. This is represented by the PPR process and activity diagrams for each resource. Input and output values could then be stored within the PPR products. Thus, by better integrating these concepts into the engineering approach of the RAMI Toolbox, its acceptability could be strongly enhanced.

However, a significant benefit of providing a reference architecture is the development, evaluation, and comparison of multiple architecture candidates, such as those presented within this thesis. To achieve a real-world assessment, automatic functions to evaluate those architecture candidates and find the best possible solutions need to be available. This could be achieved by integrating model-checking tools within the RAMI 4.0 modeling framework.

To further investigate the industrial applicability of the framework, further and more sophisticated case studies need to be applied. The chosen case studies, which were prototypically implemented in the context of this thesis, are superficial and mainly dealt with investigating the feasibility of the modeling framework. As the feasibility has been ensured by the result of this research work, future case studies need further to evaluate the modeling framework about more specific quality attributes. Thus, future enhancements of the modeling framework should thus be assessed by applying more sophisticated and actual industrial scenarios.

A follow-up project based on the outcome of this thesis deals with providing reference architectures for particular industrial domains. Thereby, more detailed modeling elements or special development guidelines are provided by the observed domain-specific peculiarities. The modeling framework must inherit abstract modeling elements for any domain, which can be instantiated, and attributes are filled with actual values. This project is already ongoing research and will be achieved in cooperation with the company partner *SIEMENS*.

Finally, the RAMI Toolbox is planned to be used in the recently started DIAMOND project, which deals with constantly exchanging engineering information across domain-specific engineering tools. Within this project, the RAMI Toolbox is used for basic engineering tasks and for creating engineering artifacts to be used in the detailed disciplines. A concrete example would be the dynamic creation of domain-specific elements and embedding within a separate metamodel. This allows each stakeholder to remain within their expertise and create a specific part-model of the industrial system.

To summarize this chapter, the proposed modeling framework provides a suitable and applicable concept for the holistic and interdisciplinary engineering of Industry 4.0 systems. By addressing a wide variety of stakeholders and ensuring the applicability of the RAMI Toolbox, most of the evaluated quality attributes achieved a positive degree of fulfillment. However, besides being an appropriate foundation for developing flexible production systems, additional fields for future work have also been identified.

Publications

Conference Publications

2023

- R. Gupta, C. Binder, A. Calà, J. Vollmar, C. Neureiter, and A. Lüder, "Enabling DSL exchange between modeling tools in a complex SoS environment," in *12th International Conference on Model and Data Engineering (MEDI 2023)*, Sousse, Tunisia, 2023.
- P. Hünecke, C. Binder, D. Hoffmann, A. Lüder, and C. Neureiter, "Facilitating FMEA investigation of industrial systems during basic engineering with RAMI 4.0," in *IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA 2023)*, Sinaia, Romania, 2023.
- H. Waclawek, G. Schäfer, C. Binder, E. Hirsch and S. Huber, "Digital Twins of Business Processes as Enablers for IT/OT Integration," in *IEEE 21st International Conference on Industrial Informatics (INDIN 2023)*, Lemgo, Germany, 2023.
- C. Binder, P. Hünecke, C. Neureiter, and A. Lüder, "Towards flexible production systems engineering according to RAMI 4.0 by utilizing PPR notation," in *IEEE 21st International Conference on Industrial Informatics (INDIN 2023)*, Lemgo, Germany, 2023.
- C. Binder, A. Calà, J. Vollmar, C. Neureiter, and A. Lüder, "Model-driven Engineering of Flexible Production Systems according to RAMI 4.0," in *IEEE 21st International Conference on Industrial Informatics (INDIN)*, Lemgo, Germany, 2023.
- C. Binder, A. Calà, J. Vollmar, C. Neureiter, and A. Lüder, "From Model to Implementation: Engineering of Flexible Production Systems with RAMI 4.0," in *IEEE 32nd International Symposium on Industrial Electronics (ISIE 2023)*, Helsinki, Finland, 2023.

2022

- C. Binder, A. Calà, J. Vollmar, C. Neureiter, and A. Lüder, “Towards Round-trip Engineering to evolve Complex Production Systems by utilizing AutomationML,” in *27th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2022)*, Stuttgart, Germany, 2022.
- C. Binder, C. Neureiter, and A. Lüder, “A bi-directional Interface enabling cross-disciplinary Engineering with RAMI 4.0 and AutomationML,” in *27th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2022)*, Stuttgart, Germany, 2022.
- C. Binder, A. Calà, J. Vollmar C. Neureiter, and A. Lüder, “RAMI Toolbox: A Framework enabling Model-based Systems Engineering of complex Flexible Production Systems,” in *13th Complex Systems Design & Management conference (CSD&M 2022)*, Paris, France, 2022.

2021

- C. Binder, A. Agic, C. Neureiter, and A. Lüder, “Applying model-based Co-Simulation on modular Production Units in Complex Automation Systems,” in *7th IEEE International Symposium on Systems Engineering (ISSE 2021)*, Vienna, Austria, 2021.
- C. Binder, G. Lastro, C. Neureiter, and A. Lüder, “Enabling model-based engineering of service-oriented Architectures within complex industrial Systems,” in *7th IEEE International Symposium on Systems Engineering (ISSE 2021)*, Vienna, Austria, 2021.
- C. Binder, A. Calà, J. Vollmar, C. Neureiter, and A. Lüder, “Automated Model Transformation in modeling Digital Twins of Industrial Internet-of-Things Applications utilizing AutomationML,” in *IEEE 26th International Conference on Emerging Technologies and Factory Automation (ETFA 2021)*, Västeras, Sweden, 2021.
- C. Binder, W. Leitner, O. Jöbstl, L. Mair, C. Neureiter, and A. Lüder, “Utilizing an Enterprise Architecture Framework for Model-Based Industrial Systems Engineering,” in *IEEE 19th International Conference on Industrial Informatics (INDIN 2021)*, Palma de Mallorca, Spain, 2021.
- C. Binder, K. Polanec, B. Brankovic, C. Neureiter, G. Lastro, and A. Lüder, “Enabling model-based requirements engineering in a complex industrial System of Systems environment,” in *IEEE 26th International Conference on Emerging Technologies and Factory Automation (ETFA 2021)*, Västeras, Sweden, 2021.

2020

- C. Binder, K. Polanec, F. Schweiberer, C. Neureiter, G. Lastro, and A. Lüder, “Using a model-based engineering approach for developing Industrial Internet of Things applications,” in *6th IEEE International Symposium on Systems Engineering (ISSE 2020)*, Vienna, Austria, 2020.
- C. Binder, B. Brankovic, C. Neureiter, and A. Lüder, “Lessons Learned from developing Industrial Applications according to RAMI 4.0 by applying Model-Based Systems Engineering,” in *25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2020)*, Vienna, Austria, 2020.
- C. Binder, M. Fischinger, K. Polanec, J. Gross, C. Neureiter, and G. Lastro, “Towards a Tool-Based Approach for Dynamically Generating Co-Simulation Scenarios based on complex Smart Grid System Architectures,” in *15th IEEE International Conference on System of Systems Engineering (SoSE 2020)*, Budapest, Hungary, 2020.
- C. Binder, D. Draxler, C. Neureiter, and G. Lastro, “Using a model-based engineering approach for developing Industrial Internet of Things applications,” in *3rd IEEE International Conference on Industrial Cyber-Physical Systems (ICPS 2020)*, Tampere, Finland, 2020.

2019

- C. Binder, D. Draxler, C. Neureiter, and G. Lastro, “Towards a Model-Centric Approach for developing Functional Architectures in Industry 4.0 Systems,” in *5th IEEE International Symposium on Systems Engineering (ISSE 2019)*, Edinburgh, Scotland, UK, 2019.
- C. Binder, J. Gross, C. Neureiter, and G. Lastro, “Investigating Emergent Behavior caused by Electric Vehicles in the Smart Grid using Co-Simulation,” in *14th Annual Conference System of Systems Engineering (SoSE 2019)*, Anchorage, Alaska, USA, 2019.

2018

- C. Binder, C. Neureiter, G. Lastro, M. Uslar, and P. Lieber, “Towards a Standards-Based Domain Specific Language for Industry 4.0 Architectures,” in *9th Complex Systems Design & Management conference (CSD&M 2018)*, Paris, France, 2018.

Journal Publications

2023

- C. Binder, A. Cala, J. Vollmar C. Neureiter, and A. Lüder, “A System Architecture Modeling Framework utilizing Model-based Systems Engineering for evolving cyber-physical Production Systems,” *Transactions on Cyber-Physical Systems*, ACM, 2023, submitted.

2022

- C. Binder, C. Neureiter, and A. Lüder, “Towards a domain-specific Information Architecture enabling the investigation and optimization of flexible Production Systems by utilizing Artificial Intelligence,” *The International Journal of Advanced Manufacturing Technology*, Springer, 2022.
- C. Neureiter and C. Binder, “A Domain-Specific, Model Based Systems Engineering Approach for Cyber-Physical Systems,” *Systems, vol. 10, Iss. 2*, 2022.

2021

- C. Binder, C. Neureiter, and A. Lüder, “Towards a Domain-Specific Approach Enabling Tool-Supported Model-Based Systems Engineering of Complex Industrial Internet-of-Things Applications,” *Systems, vol. 9, Iss. 2*, 2021.

2019

- C. Binder, M. Fischinger, L. Altenhuber, D. Draxler, G. Lastro, and C. Neureiter, “Enabling architecture based Co-Simulation of complex Smart Grid applications,” *Energy Informatics 2019, vol. 2, Iss. 1*, Springer, 2019.
- C. Binder, C. Neureiter, and G. Lastro “Towards a Model-Driven Architecture Process for Developing Industry 4.0 Applications,” *International Journal of Modeling and Optimization, vol. 10, Iss. 2*, 2019.

References

Literature

Aizenbud-Reshef, N., Nolan, B. T., Rubin, J., & Shaham-Gafni, Y. (2006). Model traceability. *IBM Systems Journal*, *45*(3), 515–526 (cit. on p. 204).

Akundi, A., Lopez, V., & Tseng, T.-L. B. (2021). Identifying the thematic trends of model based systems engineering in manufacturing and production engineering domains. *2021 IEEE International Systems Conference (SysCon)*, 1–4 (cit. on p. 55).

Al-Jaroodi, J., Mohamed, N., & Jawhar, I. (2018). A service-oriented middleware framework for manufacturing Industry 4.0. *ACM SIGBED Review*, *15*(5), 29–36 (cit. on p. 54).

Antunes, G., Barateiro, J., Becker, C., Borbinha, J., & Vieira, R. (2011). Modeling contextual concerns in enterprise architecture. *2011 IEEE 15th International Enterprise Distributed Object Computing Conference Workshops*, 3–10 (cit. on pp. 121, 123).

Anwar, M. W., Azam, F., Khan, M. A., & Butt, W. H. (2019). The applications of model driven architecture (mda) in wireless sensor networks (wsn): Techniques and tools. *Future of Information and Communication Conference*, 14–27 (cit. on p. 57).

Arantes, M., Bonnard, R., Mattei, A. P., & de Saqui-Sannes, P. (2018). General architecture for data analysis in Industry 4.0 using sysml and model based system engineering. *2018 Annual IEEE International Systems Conference (SysCon)*, 1–6 (cit. on p. 55).

Ashby, W. R. (1991). Principles of the self-organizing system. *Facets of systems science* (pp. 521–536). Springer. (Cit. on p. 17).

Atkinson, C., Tunjic, C., & Möller, T. (2015). Fundamental realization strategies for multi-view specification environments. *2015 IEEE 19th International Enterprise Distributed Object Computing Conference*, 40–49 (cit. on pp. 57, 58).

AutomationML consortium. (2014). *Whitepaper automationml part 1 - architecture and general requirements* (tech. rep.). (Cit. on pp. 71, 158).

Avizienis, A., Laprie, J.-C., & Randell, B. (2001). Fundamental concepts of dependability. *Department of Computing Science Technical Report Series* (cit. on p. 204).

Ballejos, L. C., Gonnet, S. M., & Montagna, J. M. (2008). A stakeholder model for interorganizational information systems. *International Working Conference on Requirements Engineering: Foundation for Software Quality*, 73–87 (cit. on pp. 101, 121).

Barbie, A., Hasselbring, W., Pech, N., Sommer, S., Flögel, S., & Wenzhöfer, F. (2020). Prototyping autonomous robotic networks on different layers of rami 4.0 with digital twins. *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 1–6 (cit. on p. 56).

Basl, J., & Doucek, P. (2019). A metamodel for evaluating enterprise readiness in the context of Industry 4.0. *Information*, *10*(3), 89 (cit. on p. 31).

Behnert, A.-K., Rinker, F., Lüder, A., & Biffl, S. (2021). Migrating engineering tools towards an automationml-based engineering pipeline. *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, 1–7 (cit. on p. 88).

Bentley, J. L. (1980). Multidimensional divide-and-conquer. *Communications of the ACM*, *23*(4), 214–229 (cit. on p. 28).

Berardinelli, L., Biffl, S., Lüder, A., Mätzler, E., Mayerhofer, T., Wimmer, M., & Wolny, S. (2016). Cross-disciplinary engineering with automationml and sysml. *Automatisierungstechnik*, *64*(4), 253–269 (cit. on pp. 72, 160).

Bergenthal, J. (2011). Final report model based engineering (mbe) subcommittee. *NDIA Systems Engineering Division-M&S Committee* (cit. on p. 22).

Beuche, D., Schulze, M., & Duvigneau, M. (2016). When 150% is too much: Supporting product centric viewpoints in an industrial product line. *Proceedings of the 20th International Systems and Software Product Line Conference*, 262–269 (cit. on p. 119).

Bevan, N., Kirakowskib, J., & Maissela, J. (1991). What is usability. *Proceedings of the 4th International Conference on HCI* (cit. on p. 203).

Bianchi, T., Santos, D. S., & Felizardo, K. R. (2015). Quality attributes of systems-of-systems: A systematic literature review. *2015 IEEE/ACM 3rd International Workshop on Software Engineering for Systems-of-Systems*, 23–30 (cit. on pp. 203, 204).

Biffl, S., Lüder, A., & Gerhard, D. (2017). *Multi-disciplinary engineering for cyber-physical production systems: Data models and software solutions for handling complex engineering projects*. Springer. (Cit. on pp. 8, 43, 44, 124).

Biffl, S., Meixner, K., Winkler, D., & Lüder, A. (2021). Towards efficient asset-based configuration management with a ppr asset directory. *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1–4 (cit. on p. 59).

Binder, C., Fischinger, M., Altenhuber, L., Draxler, D., Lastro, G., & Neureiter, C. (2019). Enabling architecture based co-simulation of complex smart grid applications. *Energy Informatics 2019*, *2*(1), 1–12 (cit. on p. 149).

Bitkom, VDMA, & ZVEI. (2015). *Umsetzungsstrategie Industrie 4.0: Ergebnisbericht der Plattform Industrie 4.0* (tech. rep.). Berlin, Frankfurt am Main. (Cit. on pp. 8, 46–49).

Blanchard, B. S., & Fabrycky, W. J. (2011). *Systems engineering and analysis*. Prentice Hall. (Cit. on p. 15).

Boardman, J., & Sauser, B. (2006). The meaning of system of systems. *2006 IEEE/SMC International Conference on System of Systems Engineering*, 118–123 (cit. on pp. 17, 18).

Bonanni, M., Chiti, F., Fantacci, R., & Pierucci, L. (2021). Dynamic control architecture based on software defined networking for the internet of things. *Future Internet*, *13*(5), 113 (cit. on p. 108).

Borky, J. M., & Bradley, T. H. (2018). *Effective model-based systems engineering*. Springer. (Cit. on p. 23).

Bowen, D. J., Kreuter, M., Spring, B., Cofta-Woerpel, L., Linnan, L., Weiner, D., Bakken, S., Kaplan, C. P., Squiers, L., Fabrizio, C., et al. (2009). How we design feasibility studies. *American journal of preventive medicine*, *36*(5), 452–457 (cit. on pp. 202, 203).

Brambilla, M., Cabot, J., & Wimmer, M. (2017). Model-driven software engineering in practice. *Synthesis lectures on software engineering*, *3*(1), 1–207 (cit. on pp. 20, 21).

Brankovic, B., Binder, C., Draxler, D., Neureiter, C., & Lastro, G. (2020). Towards a cross-domain modeling approach in system-of-systems architectures. *Complex Systems Design & Management*, 164–175 (cit. on pp. 95, 116).

Brettel, M., Friederichsen, N., Keller, M., & Rosenberg, M. (2014). How virtualization, decentralization and network building change the manufacturing landscape: An Industry 4.0 perspective. *International Journal of Mechanical, Industrial Science and Engineering*, *8*(1), 37–44 (cit. on p. 6).

Brown, A. W. (2004). Model driven architecture: Principles and practice. *Software and systems modeling*, *3*(4), 314–327 (cit. on p. 21).

Brusa, E. (2018). Synopsis of the mbse, lean and smart manufacturing in the product and process design for an assessment of the strategy“ Industry 4.0”. *CIISE*, 21–30 (cit. on p. 8).

Büscher, M., Claassen, A., Kube, M., Lehnhoff, S., Piech, K., Rohjans, S., Scherfke, S., Steinbrink, C., Velasquez, J., Tempez, F., et al. (2014). Integrated smart grid simulations for generic automation architectures with rt-lab and mosaik. *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 194–199 (cit. on p. 45).

Calà, A. (2019). *A novel migration approach towards decentralized automation in cyber-physical production systems* (Doctoral dissertation). Otto von Guericke University Library, Magdeburg, Germany. (Cit. on pp. 2, 9, 114).

Calà, A., Lüder, A., Cachada, A., Pires, F., Barbosa, J., Leitão, P., & Gepp, M. (2017). Migration from traditional towards cyber-physical production systems. *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, 147–152 (cit. on p. 9).

Carlock, P. G., & Fenton, R. E. (2001). System of systems (sos): Enterprise systems engineering for information-intensive organizations. *Systems engineering*, *4*(4), 242–261 (cit. on p. 16).

Carvajal Soto, J., Tavakolizadeh, F., & Gyulai, D. (2019). An online machine learning framework for early detection of product failures in an Industry 4.0 context. *International Journal of Computer Integrated Manufacturing*, *32*(4-5), 452–465 (cit. on p. 146).

Chakrabarti, S. (2022). Detailed engineering. *Project engineering primer for chemical engineers* (pp. 109–134). Springer. (Cit. on p. 62).

Chen, W. K. (2004). *The electrical engineering handbook*. Elsevier. (Cit. on p. 62).

Claude Jr, R. M., & Horne, D. A. (1992). Restructuring towards a service orientation: The strategic challenges. *International Journal of Service Industry Management*, *3*(1), 25–38 (cit. on p. 1).

Cloutier, R., Muller, G., Verma, D., Nilchiani, R., Hole, E., & Bone, M. (2010). The concept of reference architectures. *Systems Engineering*, *13*(1), 14–27 (cit. on pp. 32, 33).

Conboy, K., Gleasure, R., & Cullina, E. (2015). Agile Design Science Research. *New Horizons in Design Science: Broadening the Research Agenda*, 168–180 (cit. on pp. 78, 79).

Contreras, J. D., Garcia, J. I., & Pastrana, J. D. (2017). Developing of Industry 4.0 applications. *International Journal of Online Engineering*, *13*(10), 30–47 (cit. on pp. 8, 54).

Dall'Ora, N., Alamin, K., Fraccaroli, E., Poncino, M., Quaglia, D., & Vinco, S. (2021). Digital transformation of a production line: Network design, online data collection and energy monitoring. *IEEE Transactions on Emerging Topics in Computing*, *10*(01), 46–59 (cit. on p. 2).

Dänekas, C., Neureiter, C., Rohjans, S., Uslar, M., & Engel, D. (2014). Towards a model-driven-architecture process for Smart Grid projects. *Digital enterprise design & management* (pp. 47–58). Springer. (Cit. on p. 91).

Deeba, F., Kun, S., Shaikh, M., Dharejo, F. A., Hayat, S., & Suwansrikham, P. (2018). Data transformation of uml diagram by using model driven architecture. *3rd IEEE International Conference on Cloud Computing and Big Data Analysis (ICCBDA)*, 300–303 (cit. on p. 57).

DeLaurentis, D. (2005). Understanding transportation as a system-of-systems design problem. *43rd AIAA Aerospace Sciences Meeting and Exhibit* (cit. on pp. 18, 19).

Delligatti, L. (2013). *Sysml distilled: A brief guide to the systems modeling language*. Addison-Wesley. (Cit. on p. 25).

de Santos, F. J. N., & Villalonga, S. G. (2015). Exploiting local clouds in the internet of everything environment. *Parallel, Distributed and Network-Based Processing (PDP), 2015 23rd Euromicro International Conference on*, 296–300 (cit. on p. 5).

Dikhanbayeva, D., Shaikholla, S., Suleiman, Z., & Turkyilmaz, A. (2020). Assessment of Industry 4.0 maturity models by design principles. *Sustainability*, *12*(23), 1–22 (cit. on p. 7).

Dragicevic, N., Ullrich, A., Tsui, E., & Gronau, N. (2020). A conceptual model of knowledge dynamics in the Industry 4.0 Smart Grid scenario. *Knowledge Management Research & Practice*, *18*(2), 199–213 (cit. on p. 145).

Drath, R. (2012). Let's talk automationml what is the effort of automationml programming? *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012)*, 1–8 (cit. on p. 161).

Drath, R. (2021). *Automationml: A practical guide*. Walter de Gruyter GmbH & Co KG. (Cit. on pp. 69, 71).

Drath, R., & Horch, A. (2014). Industrie 4.0: Hit or hype? *IEEE industrial electronics magazine*, *8*(2), 56–58 (cit. on pp. 4, 5).

El Kho, F. Z., & Aknin, N. (2019). Thing-based service-oriented architecture for Industry 4.0. *International Conference on Artificial Intelligence and Symbolic Computation*, 289–300 (cit. on p. 54).

Emery, D., & Hilliard, R. (2009). Every architecture description needs a framework: Expressing architecture frameworks using ISO/IEC 42010. *2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture*, 31–40 (cit. on p. 35).

Erraissi, A., & Belangour, A. (2020). An approach based on model driven engineering for big data visualization in different visual modes. *International Journal of Scientific & Technology Research*, *9*(1), 198–206 (cit. on p. 57).

Fan, Y., Yang, J., Chen, J., Hu, P., Wang, X., Xu, J., & Zhou, B. (2021). A digital-twin visualized architecture for flexible manufacturing system. *Journal of Manufacturing Systems*, *60*, 176–201 (cit. on p. 56).

Fischinger, M., Neureiter, C., Binder, C., Egger, N., & Renoth, M. (2019). Fredosar: Towards a security-aware open system architecture framework supporting model based systems engineering. *SMARTGREENS*, 108–115 (cit. on p. 152).

Förster, R., & Förster, A. (2018). Einteilung der fertigungsverfahren nach DIN 8580. *Einführung in die fertigungstechnik* (pp. 23–136). Springer. (Cit. on p. 59).

Fowler, M. (2005). Language workbenches: The killer-app for domain specific languages (cit. on p. 32).

Friedenthal, S., Moore, A., & Steiner, R. (2014). *A practical guide to sysml: The systems modeling language*. Morgan Kaufmann. (Cit. on p. 37).

Friedenthal, S., & Oster, C. (2017). *Architecting spacecraft with sysml: A model-based systems engineering approach*. CreateSpace Independent Publishing Platform. (Cit. on p. 3).

Gaikwad, S., & Kulkarni, H. (2014). Service quality improvement by kaizen through makigami analysis. *Proceedings of International Conference on Electrical, Electronics, Computer Science and Mechanical Engineering*, 53–58 (cit. on p. 96).

Gausemeier, J., & Moehringer, S. (2002). VDI 2206-a new guideline for the design of mechatronic systems. *IFAC Proceedings Volumes*, *35*(2), 785–790 (cit. on pp. 11, 60, 61).

Gerber, A., le Roux, P., Kearney, C., & van der Merwe, A. (2020). The zachman framework for enterprise architecture: An explanatory is theory. *Conference on e-Business, e-Services and e-Society*, 383–396 (cit. on p. 65).

Ghobakhloo, M. (2018). The future of manufacturing industry: A strategic roadmap toward Industry 4.0. *Journal of Manufacturing Technology Management*, *29*(6), 910–936 (cit. on pp. 1, 7).

Glasmeier, A. (1991). Technological discontinuities and flexible production networks: The case of switzerland and the world watch industry. *Research policy*, *20*(5), 469–485 (cit. on p. 2).

Golfarelli, M., & Rizzi, S. (2020). A model-driven approach to automate data visualization in big data analytics. *Information Visualization*, *19*(1), 24–47 (cit. on p. 57).

Gorecky, D., Schmitt, M., Loskyll, M., & Zühlke, D. (2014). Human-machine-interaction in the Industry 4.0 era. *2014 12th IEEE International Conference on Industrial Informatics (INDIN 2014)*, 289–294 (cit. on p. 6).

Grangel Gonzalez, I. (2019). *A knowledge graph based integration approach for Industry 4.0* (Doctoral dissertation). Universitäts-und Landesbibliothek Bonn. (Cit. on p. 55).

Grangel-González, I., Halilaj, L., Coskun, G., Auer, S., Collarana, D., & Hoffmeister, M. (2016). Towards a semantic administrative shell for Industry 4.0 components. *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)*, 230–237 (cit. on p. 55).

Grote, K.-H., & Antonsson, E. K. (2009). *Springer handbook of mechanical engineering* (Vol. 10). Springer. (Cit. on p. 62).

Gutierrez-Guerrero, J. M., & Holgado-Terriza, J. A. (2019). Automatic configuration of opc ua for industrial internet of things environments. *Electronics*, *8*(6), 600 (cit. on p. 151).

Haberfellner, R., de Weck, O., Fricke, E., & Vössner, S. (2015). *Systems engineering - grundlagen und anwendung* (13th ed.). Orell Füssli. (Cit. on pp. 2, 3).

Hacks, S., & Katsikeas, S. (2021). Towards an ecosystem of domain specific languages for threat modeling. *International Conference on Advanced Information Systems Engineering*, 3–18 (cit. on p. 33).

Haoues, M., Sellami, A., Ben-Abdallah, H., & Cheikhi, L. (2017). A guideline for software architecture selection based on ISO 25010 quality related characteristics. *International Journal of System Assurance Engineering and Management*, *8*(2), 886–909 (cit. on p. 203).

Harrison, R. (2018). *Togaf (r) 9 foundation study guide*. Van Haren. (Cit. on pp. 40, 127).

Hay, D. C. (1997). The zachman framework: An introduction. *The Data Administration Newsletter*, (1) (cit. on pp. 64, 65).

Heinen, T., Peter, K., Erlach, K., Nyhuis, P., Lanza, G., & Westkämper, E. (2010). Zukunftsthemen der fabrikplanung. *Zeitschrift für wirtschaftlichen Fabrikbetrieb*, *105*(5), 405–409 (cit. on p. 60).

Henzinger, T. A. (2000). The theory of hybrid automata. *Verification of digital and hybrid systems* (pp. 265–292). Springer. (Cit. on p. 45).

Hermann, M., Pentek, T., & Otto, B. (2016). Design principles for industrie 4.0 scenarios. *2016 49th Hawaii International Conference on System Sciences (HICSS)*, 3928–3937 (cit. on p. 5).

Herzog, J., Röpke, H., & Lüder, A. (2020). Allocation of pprs for the plant planning in the final automotive assembly. *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, *1*, 813–820 (cit. on p. 59).

Hettel, T., Lawley, M., & Raymond, K. (2008). Model synchronisation: Definitions for round-trip engineering. *International Conference on Theory and Practice of Model Transformations*, 31–45 (cit. on p. 24).

Hevner, A., & Chatterjee, S. (2010). Design science research in information systems. *Design research in information systems* (pp. 9–22). Springer. (Cit. on pp. 4, 75–77, 164, 216).

Hilliard, R. (2000). Ieee 1471-2000: Recommended practice for architectural description for software-intensive systems. *IEEE*, 1–30 (cit. on pp. 26, 33).

Hunkeler, U., Truong, H. L., & Stanford-Clark, A. (2008). Mqtt-s—a publish/subscribe protocol for wireless sensor networks. *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*, 791–798 (cit. on p. 151).

Iglesias, A., Sagardui, G., & Arellano, C. (2019). Industrial cyber-physical system evolution detection and alert generation. *Applied Sciences*, *9*(8), 1586 (cit. on p. 1).

Indri, M., Grau, A., & Ruderman, M. (2018). Guest editorial special section on recent trends and developments in Industry 4.0 motivated robotic solutions. *IEEE Transactions on Industrial Informatics*, *14*(4), 1677–1680 (cit. on p. 84).

International Electrotechnical Commission. (2001). *IEC 61512: Batch control* (Standard). (Cit. on p. 49).

International Electrotechnical Commission. (2016a). *IEC 62264: Enterprise-control system integration* (Standard). (Cit. on p. 49).

International Electrotechnical Commission. (2016b). *IEC 62890: Life-cycle management for systems and products used in industrial-process measurement, control and automation* (Standard). (Cit. on p. 49).

International Organization for Standardization. (2011). *ISO/IEC/IEEE 42010: Systems and software engineering – architecture description* (Standard). (Cit. on p. 34).

International Organization for Standardization. (2015). *ISO 15288:2015 Systems engineering - System life cycle processes* (Standard). (Cit. on p. 39).

Jabraeil Jamali, M. A., Bahrami, B., Heidari, A., Allahverdizadeh, P., & Norouzi, F. (2020). Iot architecture. *Towards the Internet of Things*, 9–31 (cit. on p. 108).

Jeon, B., Yoon, J.-S., Um, J., & Suh, S.-H. (2020). The architecture development of Industry 4.0 compliant smart machine tool system (SMTS). *Journal of Intelligent Manufacturing*, *31*(8), 1837–1859 (cit. on p. 56).

Kandé, M. M., Crettaz, V., Strohmeier, A., & Sendall, S. (2002). Bridging the gap between IEEE 1471, an architecture description language, and uml. *Software and Systems Modeling*, *1*(2), 113–129 (cit. on p. 34).

Kazman, R., Abowd, G., Bass, L., & Clements, P. (1996). Scenario-based analysis of software architecture. *IEEE software*, *13*(6), 47–55 (cit. on pp. 79–81).

Kempa, M., & Mann, Z. A. (2005). Model driven architecture. *Informatik-Spektrum*, *28*(4), 298–302 (cit. on pp. 29, 30).

Kerzhner, A. A., & Paredis, C. J. (2009). Using domain specific languages to capture design synthesis knowledge for model-based systems engineering. *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 1399–1409 (cit. on p. 33).

Khan, W. Z., Rehman, M., Zangoti, H. M., Afzal, M. K., Armi, N., & Salah, K. (2020). Industrial internet of things: Recent advances, enabling technologies and open challenges. *Computers & Electrical Engineering*, *81*, 106522 (cit. on p. 2).

Kirsch, L. J. (1996). The management of complex tasks in organizations: Controlling the systems development process. *Organization science*, *7*(1), 1–21 (cit. on p. 109).

Klecun, E., & Cornford, T. (2005). A critical approach to evaluation. *European Journal of Information Systems*, *14*(3), 229–243 (cit. on p. 165).

Kleppe, A. (2008). *Software language engineering: Creating domain-specific languages using metamodels*. Pearson Education. (Cit. on p. 24).

Kleppe, A. G., Warmer, J. B., & Bast, W. (2003). *MDA explained: the model driven architecture: practice and promise*. Addison-Wesley Professional. (Cit. on p. 28).

Koh, L., Orzes, G., & Jia, F. J. (2019). The fourth industrial revolution (Industry 4.0): Technologies disruption on operations and supply chain management. *International Journal of Operations & Production Management*, *39*(6/7/8), 817–828 (cit. on p. 8).

Kosek, A. M., Lünsdorf, O., Scherfke, S., Gehrke, O., & Rohjans, S. (2014). Evaluation of smart grid control strategies in co-simulation—integration of ipsys and mosaik. *2014 Power Systems Computation Conference*, 1–7 (cit. on p. 45).

Kruchten, P. B. (1995). The 4+1 view model of architecture. *IEEE software*, *12*(6), 42–50 (cit. on p. 35).

Krüger, J., Mahmood, W., & Berger, T. (2020). Promote-pl: A round-trip engineering process model for adopting and evolving product lines. *Proceedings of the 24th*

ACM Conference on Systems and Software Product Line: Volume A, 1–12 (cit. on p. 24).

Kulcsár, G., Koltai, K., Tanyi, S., Péceli, B., Horváth, Á., Micskei, Z., & Varga, P. (2020). From models to management and back: Towards a system-of-systems engineering toolchain. *2020 IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 1–6 (cit. on p. 43).

Kulcsár, G., Tatara, M. S., & Montori, F. (2020). Toolchain modeling: Comprehensive engineering plans for Industry 4.0. *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, 4541–4546 (cit. on p. 42).

Łabiak, G., & Bazydło, G. (2018). Model driven architecture approach to logic controller design. *AIP Conference Proceedings*, *2040*(1) (cit. on p. 57).

Lake, J. G. (1997). Thoughts about life cycle phases: How a system is developed incrementally. *INCOSE International Symposium*, *7*(1), 597–603 (cit. on p. 39).

Law, A. M., Kelton, W. D., & Kelton, W. D. (2000). *Simulation modeling and analysis* (Vol. 3). McGraw-Hill New York. (Cit. on p. 44).

Li, H., Zou, M., Hogrefe, G., Ryashentseva, D., Sollfrank, M., Koltun, G., & Vogel-Heuser, B. (2019). Application of a multi-disciplinary design approach in a mechatronic engineering toolchain. *at-Automatisierungstechnik*, *67*(3), 246–269 (cit. on p. 44).

Li, Z., Sim, C. H., & Low, M. Y. H. (2006). A survey of emergent behavior and its impacts in agent-based systems. *2006 4th IEEE international conference on industrial informatics*, 1295–1300 (cit. on p. 146).

Lightsey, B. (2001). *Systems engineering fundamentals* (tech. rep.). DTIC Document. (Cit. on pp. 15, 23).

Lin, S.-W., Miller, B., Durand, J., Joshi, R., Didier, P., Chigani, A., Torenbeek, R., Duggal, D., Martin, R., Bleakley, G., et al. (2015). Industrial Internet Reference Architecture (IIRA). *Industrial Internet Consortium (IIC), Tech. Rep* (cit. on p. 67).

Linkin Park. (2010). A Thousand Suns: Track 8 - Waiting for the End. (Cit. on p. 220).

Lins, T., & Oliveira, R. A. R. (2020). Cyber-physical production systems retrofitting in context of Industry 4.0. *Computers & industrial engineering*, *139*, 106193 (cit. on p. 56).

Liu, B., Glock, T., Betancourt, V. P., Kern, M., Sax, E., & Becker, J. (2020). Model driven development process for a service-oriented Industry 4.0 system. *2020 9th International Conference on Industrial Technology and Management (ICITM)*, 78–83 (cit. on p. 54).

Lopez, M. M., Bonito-Oliva, A., Valente, P., Boutboul, T., de Sousa, P. C., Loizaga, A., Romano, G., Aprili, P., Harrison, R., Gavouyere-Lasserre, P., et al. (2022). Factory acceptance test and delivery of the first two poloidal field coils to iter fusion facility. *IEEE Transactions on Applied Superconductivity*, *32*(6), 1–5 (cit. on p. 8).

Lu, J., Wang, G., & Törngren, M. (2019). Design ontology in a case study for cosimulation in a model-based systems engineering tool-chain. *IEEE Systems journal*, *14*(1), 1297–1308 (cit. on p. 44).

Lüder, A., Baumann, L., Behnert, A.-K., Rinker, F., & Biffl, S. (2020). Paving pathways for digitalization in engineering: Common concepts in engineering chains. *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, *1*, 1401–1404 (cit. on p. 2).

Lüder, A., Foehr, M., Hundt, L., Hoffmann, M., Langer, Y., & Frank, S. (2011). Aggregation of engineering processes regarding the mechatronic approach. *2011 IEEE Conference on Emerging Technologies & Factory Automation*, 1–8 (cit. on p. 60).

Lüder, A., Kirchheim, K., Pauly, J., Biffl, S., Rinker, F., & Waltersdorfer, L. (2019). Supporting the data model integrator in an engineering network by automating data integration. *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, *1*, 1229–1234 (cit. on pp. 72, 73).

Lüder, A., Pauly, J.-L., Rinker, F., & Biffl, S. (2019). Data exchange logistics in engineering networks exploiting automated data integration. *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 657–664 (cit. on pp. 57, 72).

Lüder, A., & Schmidt, N. (2017). Automationml in a nutshell. *Handbuch Industrie 4.0 Bd. 2* (pp. 213–258). Springer. (Cit. on p. 71).

Lüder, A., Schmidt, N., & Drath, R. (2017). Standardized information exchange within production system engineering. *Multi-disciplinary engineering for cyber-physical production systems* (pp. 235–257). Springer. (Cit. on p. 69).

Madni, A. M. (2021). Mbse testbed for rapid, cost-effective prototyping and evaluation of system modeling approaches. *Applied Sciences*, *11*(5), 2321 (cit. on p. 57).

Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering: The Journal of the International Council on Systems Engineering*, *1*(4), 267–284 (cit. on pp. 2, 18, 19).

Maier, M. W., Emery, D., & Hilliard, R. (2001). Software architecture: Introducing IEEE standard 1471. *Computer*, *34*(4), 107–109 (cit. on p. 34).

Maier, M. W., Emery, D., & Hilliard, R. (2004). ANSI/IEEE 1471 and systems engineering. *Systems engineering*, *7*(3), 257–270 (cit. on p. 34).

Mandel, C., Stürmlinger, T., Yue, C., Behrendt, M., & Albers, A. (2020). Model-based systems engineering approaches for the integrated development of product and production systems in the context of Industry 4.0. *2020 IEEE International Systems Conference (SysCon)*, 1–7 (cit. on p. 55).

Martin, R. C. (1996). The dependency inversion principle. *C++ Report*, *8*(6), 61–66 (cit. on p. 27).

Martıénez, P. L., Dintén, R., Drake, J. M., & Zorrilla, M. (2021). A big data-centric architecture metamodel for Industry 4.0. *Future Generation Computer Systems*, *125*, 263–284 (cit. on p. 31).

Mehr, R., & Lüder, A. (2019). Managing complexity within the engineering of product and production systems. *Security and quality in cyber-physical systems engineering* (pp. 57–79). Springer. (Cit. on p. 8).

Meixner, K., Lüder, A., Herzog, J., Röpke, H., & Biffl, S. (2020). Modeling expert knowledge for optimal cpps resource selection for a product portfolio. *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, *1*, 1687–1694 (cit. on p. 154).

Melouk, E. M., Rhazali, Y., & Hadi, Y. (2021). Model transformation approach according model-driven architecture from bpmn to uml up to ifml. *Advancements in model-driven architecture in software engineering* (pp. 172–191). IGI Global. (Cit. on p. 57).

Mendoza-Pitti, L., Calderón-Gómez, H., Vargas-Lombardo, M., Gómez-Pulido, J. M., & Castillo-Sequera, J. L. (2021). Towards a service-oriented architecture for the energy efficiency of buildings: A systematic review. *IEEE Access*, *9*, 26119–26137 (cit. on p. 53).

Merwe, A. v. d., Gerber, A., & Smuts, H. (2019). Guidelines for conducting design science research in information systems. *Annual Conference of the Southern African Computer Lecturers' Association*, 163–178 (cit. on pp. 77, 78).

Metallica. (1992). Metallica: Track 8 - Nothing Else Matters. (Cit. on p. 37).

Mezhuyev, V., & Samet, R. (2013). Geometrical meta-metamodel for cyber-physical modelling. *Cyberworlds (CW), 2013 International Conference on*, 89–93 (cit. on pp. 31, 91).

Molano, J. I. R., Lovelle, J. M. C., Montenegro, C. E., Granados, J., & Crespo, R. G. (2018). Metamodel for integration of internet of things, social networks, the cloud and Industry 4.0. *Journal of ambient intelligence and humanized computing*, *9*(3), 709–723 (cit. on p. 31).

Morkevicius, A., Bisikirskiene, L., & Bleakley, G. (2017). Using a systems of systems modeling approach for developing industrial internet of things applications. *2017 12th System of Systems Engineering Conference (SoSE)*, 1–6 (cit. on p. 56).

Mrugalska, B., & Wyrwicka, M. K. (2017). Towards lean production in Industry 4.0. *Procedia engineering*, *182*, 466–473 (cit. on p. 8).

Neureiter, C. (2017). *A domain-specific, model driven engineering approach for systems engineering in the smart grid*. MBSE4U - Tim Weilkiens. (Cit. on p. 109).

Neureiter, C., Engel, D., & Uslar, M. (2016). Domain specific and model based systems engineering in the smart grid as prerequesite for security by design. *Electronics*, *5*(2), 24 (cit. on p. 10).

Nuseibeh, B., & Easterbrook, S. (2000). Requirements engineering: A roadmap. *Proceedings of the Conference on the Future of Software Engineering*, 35–46 (cit. on p. 25).

O'Brien, L., Merson, P., & Bass, L. (2007). Quality attributes for service-oriented architectures. *International Workshop on Systems Development in SOA Environments (SDSOA'07: ICSE Workshops 2007)*, 3–3 (cit. on p. 81).

Orlikowski, W. J., & Robey, D. (1991). Information technology and the structuring of organizations. *Information systems research*, *2*(2), 143–169 (cit. on p. 62).

Palensky, P., Van Der Meer, A. A., Lopez, C. D., Joseph, A., & Pan, K. (2017). Cosimulation of intelligent power systems: Fundamentals, software architecture, numerics, and coupling. *IEEE Industrial Electronics Magazine*, *11*(1), 34–50 (cit. on pp. 44, 45).

Pearce, P., & Hause, M. (2012). ISO-15288, oosem and model-based submarine design (cit. on p. 40).

Peffers, K., Rothenberger, M., Tuunanen, T., & Vaezi, R. (2012). Design science research evaluation. *International Conference on Design Science Research in Information Systems*, 398–410 (cit. on pp. 165, 166).

Perrey, R., & Lycett, M. (2003). Service-oriented architecture. *2003 Symposium on Applications and the Internet Workshops, Proceedings.*, 116–119 (cit. on pp. 52, 53).

Pfrommer, J., Schleipen, M., & Beyerer, J. (2013). Pprs: Production skills and their relation to product, process, and resource. *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*, 1–4 (cit. on pp. 2, 59, 60).

Pisching, M. A., Pessoa, M. A., Junqueira, F., dos Santos Filho, D. J., & Miyagi, P. E. (2018). An architecture based on RAMI 4.0 to discover equipment to process operations required by products. *Computers & Industrial Engineering*, *125*, 574–591 (cit. on pp. 55, 56).

Place, F. N. (2000). Object management group. *mars*, *2005*, 06–12 (cit. on p. 21).

Plattform Industrie 4.0. (2016). *Struktur der Verwaltungsschale* (tech. rep.). (Cit. on pp. 51, 52).

Pohl, K., Broy, M., Daembkes, H., & Hönninger, H. (2016). *Advanced model-based engineering of embedded systems*. Springer. (Cit. on pp. 66, 67).

Pohl, K., Hönninger, H., Achatz, R., & Broy, M. (2012). *Model-based engineering of embedded systems: The SPES 2020 methodology*. Springer Science & Business Media. (Cit. on p. 67).

Pohl, K., & Rupp, C. (2015). *Requirements engineering - fundamentals, principles, and techniques*. dpunkt. verlag. (Cit. on p. 25).

Posch, T., Gerdom, M., & Birken, K. (2012). *Basiswissen Softwarearchitektur: verstehen, entwerfen, wiederverwenden*. dpunkt. verlag. (Cit. on p. 29).

Pries-Heje, J., Baskerville, R., & Venable, J. R. (2008). Strategies for design science research evaluation (cit. on p. 165).

Putman, J. (2001). *Architecting with rm-odp*. Prentice Hall Professional. (Cit. on p. 35).

Qamar, A., Herzig, S. J., & Paredis, C. J. (2013). A domain-specific language for dependency management in model-based systems engineering. *MPM@ MoDELS*, 7–16 (cit. on p. 33).

Rajkumar, R., Lee, I., Sha, L., & Stankovic, J. (2010). Cyber-physical systems: The next computing revolution. *Design Automation Conference*, 731–736 (cit. on pp. 1, 7).

Ramos, A. L., Ferreira, J. V., & Barceló, J. (2011). Model-based systems engineering: An emerging approach for modern systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*(1), 101–111 (cit. on p. 23).

Ramos, L., Loures, E., Deschamps, F., & Venâncio, A. (2020). Systems evaluation methodology to attend the digital projects requirements for Industry 4.0. *International Journal of Computer Integrated Manufacturing*, *33*(4), 398–410 (cit. on p. 203).

Reis, J. Z., & Gonçalves, R. F. (2018). The role of internet of services (ios) on Industry 4.0 through the service oriented architecture (soa). *IFIP International Conference on Advances in Production Management Systems*, 20–26 (cit. on p. 53).

Rementsov, A., & Lukinov, V. (2020). The new paradigm of virtual enterprises and application of flexible production systems. *E3S Web of Conferences*, *210*, 13024 (cit. on p. 56).

Resman, M., Pipan, M., Šimic, M., & Herakovič, N. (2019). A new architecture model for smart manufacturing: A performance analysis and comparison with the RAMI 4.0 reference model. *Adv. Prod. Eng. Manag*, *14*(2), 153–165 (cit. on p. 49).

Rother, M., & Shook, J. (2003). *Learning to see: Value stream mapping to add value and eliminate muda*. Lean Enterprise Institute. (Cit. on pp. 96, 173).

Rozanski, N., & Woods, E. (2011). *Software systems architecture: Working with stakeholders using viewpoints and perspectives*. Addison-Wesley. (Cit. on p. 26).

Salado, A., & Wach, P. (2019). Constructing true model-based requirements in sysml. *Systems*, *7*(2), 19 (cit. on p. 25).

Salman, A. J., Al-Jawad, M., & Al Tameemi, W. (2021). Domain-specific languages for iot: Challenges and opportunities. *IOP Conference Series: Materials Science and Engineering*, *1067*(1), 012133 (cit. on p. 33).

Sarjoughian, H. S. (2006). Model composability. *Proceedings of the 2006 Winter Simulation Conference*, 149–158 (cit. on p. 205).

Saucedo Martıénez, J. A., & Noriega, C. R. (2020). Literature review: Evaluation of the feasibility of implementing Industry 4.0 technologies in the intralogistic processes of the logistics operators of the department of the atlantic, a look towards the continuous improvement of organizational efficiency. *Data Analysis and Optimization for Engineering and Computing Problems*, 125–141 (cit. on p. 202).

Schleipen, M., & Drath, R. (2009). Three-view-concept for modeling process or manufacturing plants with automationml. *2009 IEEE Conference on Emerging Technologies & Factory Automation*, 1–4 (cit. on pp. 58, 59).

Schloegl, F., Rohjans, S., Lehnhoff, S., Velasquez, J., Steinbrink, C., & Palensky, P. (2015). Towards a classification scheme for co-simulation approaches in energy systems. *2015 International Symposium on Smart Electric Distribution Systems and Technologies (EDST)*, 516–521 (cit. on p. 44).

Schluse, M., Priggemeyer, M., Atorf, L., & Rossmann, J. (2018). Experimentable digital twins—streamlining simulation-based systems engineering for Industry 4.0. *IEEE Transactions on industrial informatics*, *14*(4), 1722–1731 (cit. on p. 55).

Schmidt, D. C. (2006). Model-driven engineering. *Computer-IEEE Computer Society-*, *39*(2), 25–31 (cit. on p. 22).

Schnicke, F., Kuhn, T., & Antonino, P. O. (2020). Enabling Industry 4.0 service-oriented architecture through digital twins. *European Conference on Software Architecture*, 490–503 (cit. on p. 53).

Schuh, G., Potente, T., Wesch-Potente, C., & Hauptvogel, A. (2013). Sustainable increase of overhead productivity due to cyber-physical-systems. *2013 11th Global Conference on Sustainable Manufacturing (GCSM 2013)*, 332–335 (cit. on p. 6).

Schumacher, A., Sihn, W., & Erol, S. (2016). Automation, digitization and digitalization and their implications for manufacturing processes. *Innovation and Sustainability Conference Bukarest*, 1–5 (cit. on p. 57).

Schütte, S., Scherfke, S., & Sonnenschein, M. (2012). Mosaik-smart grid simulation api. *Proceedings of SMARTGREENS*, 14–24 (cit. on p. 148).

Schütte, S., Scherfke, S., & Tröschel, M. (2011). Mosaik: A framework for modular simulation of active components in smart grids. *2011 IEEE First International Workshop on Smart Grid Modeling and Simulation (SGMS)*, 55–60 (cit. on pp. 45, 147).

Selic, B. (2003). The pragmatics of model-driven development. *IEEE software*, *20*(5), 19–25 (cit. on p. 22).

Sharpe, R., van Lopik, K., Neal, A., Goodall, P., Conway, P. P., & West, A. A. (2019). An industrial evaluation of an Industry 4.0 reference architecture demonstrating the need for the inclusion of security and human components. *Computers in Industry*, *108*, 37–44 (cit. on p. 55).

Sindico, A., Di Natale, M., & Sangiovanni-Vincentelli, A. (2012). An industrial system engineering process integrating model driven architecture and model based design. *International Conference on Model Driven Engineering Languages and Systems*, 810–826 (cit. on pp. 54, 159).

Sommerville, I., & Sawyer, P. (1997). *Requirements engineering: A good practice guide*. John Wiley & Sons, Inc. (Cit. on p. 25).

Sparks, G. (2009). Enterprise Architect user guide (cit. on p. 94).

Stahl, T., & Volter, M. (2006). *Model-driven software development: Technology, engineering, management*. J. Wiley & Sons. (Cit. on pp. 30–32).

Steinbrink, C., Blank-Babazadeh, M., El-Ama, A., Holly, S., Lüers, B., Nebel-Wenner, M., Ramıérez Acosta, R. P., Raub, T., Schwarz, J. S., Stark, S., et al. (2019). CPES testing with mosaik: Co-simulation planning, execution and analysis. *Applied Sciences*, *9*(5), 923 (cit. on p. 45).

Talkhestani, B. A., Jung, T., Lindemann, B., Sahlab, N., Jazdi, N., Schloegl, W., & Weyrich, M. (2019). An architecture of an intelligent digital twin in a cyber-physical production system. *at-Automatisierungstechnik*, *67*(9), 762–782 (cit. on p. 56).

Tliba, K., Penas, O., Diallo, T. M., Khalifa, R. B., Yahia, N. B., & Choley, J.-Y. (2020). Model based systems engineering approach for the improvement of manufacturing system flexibility. *2020 21st International Conference on Research and Education in Mechatronics (REM)*, 1–6 (cit. on p. 54).

Tolio, T. (2008). *Design of flexible production systems*. Springer. (Cit. on p. 54).

Unverdorben, S. (2021). *Architecture framework concept for definition of system architectures based on reference architectures within the domain manufacturing* (Doctoral dissertation). (Cit. on pp. 11, 26).

Van Deursen, A., Klint, P., Visser, J., et al. (2000). Domain-specific languages: An annotated bibliography. *Sigplan Notices*, *35*(6), 26–36 (cit. on p. 32).

Van Lamsweerde, A. (2000). Requirements engineering in the year 00: A research perspective. *Proceedings of the 22nd international conference on Software engineering*, 5–19 (cit. on p. 24).

Verein Deutscher Ingenieure. (2009). VDI richtlinie 3695–Engineering von Anlagen–Evaluieren und Optimieren des Engineerings (cit. on p. 60).

Villalonga, A., Beruvides, G., Castaño, F., & Haber, R. E. (2020). Cloud-based industrial cyber–physical system for data-driven reasoning: A review and use case on an Industry 4.0 pilot line. *IEEE Transactions on Industrial Informatics*, *16*(9), 5975–5984 (cit. on pp. 2, 56).

Vogel, O., Arnold, I., Chughtai, A., Ihler, E., Kehrer, T., Mehlig, U., & Zdun, U. (2008). *Software-Architektur: Grundlagen-Konzepte-Praxis*. Springer Science & Business Media. (Cit. on pp. 26–28, 32).

Wang, Y., Towara, T., & Anderl, R. (2017). Topological approach for mapping technologies in RAMI 4.0. *Proceedings of the World Congress on Engineering and Computer Science*, *2*, 25–27 (cit. on p. 50).

Watson, A. (2004). Introduction to model driven architecture ‘. *OMG Information Day on Integrating the Enterprise* (cit. on p. 29).

Weilkiens, T. (2011). *Systems engineering with sysml/uml: Modeling, analysis, design*. Elsevier. (Cit. on p. 20).

Weilkiens, T. (2016). *Sysmod-the systems modeling toolbox-pragmatic mbse with sysml*. MBSE4U. (Cit. on p. 96).

Weilkiens, T., Lamm, J. G., Roth, S., & Walker, M. (2015). *Model-based system architecture*. John Wiley & Sons. (Cit. on pp. 20, 38, 182).

Winkler, D., Ekaputra, F., & Biffl, S. (2016). Automationml review support in multidisciplinary engineering environments. *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1–9 (cit. on pp. 69, 70).

Wortmann, A., Barais, O., Combemale, B., & Wimmer, M. (2020). Modeling languages in Industry 4.0: An extended systematic mapping study. *Software and Systems Modeling*, *19*(1), 67–94 (cit. on p. 56).

Wymore, A. W. (2018). *Model-based systems engineering* (Vol. 3). CRC press. (Cit. on p. 20).

Xue, R., Baron, C., & Esteban, P. (2017). Optimizing product development in industry by alignment of the ISO/IEC 15288 systems engineering standard and the pmbok guide. *International Journal of Product Development*, *22*(1), 65–80 (cit. on p. 39).

Yang, C.-W., & Vyatkin, V. (2018). On automated co-simulation testing of functional requirements for distributed substation automation systems. *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, 3576–3581 (cit. on p. 44).

Yang, C.-h., Zhabelova, G., Yang, C.-W., & Vyatkin, V. (2013). Cosimulation environment for event-driven distributed controls of smart grid. *IEEE Transactions on Industrial Informatics*, *9*(3), 1423–1435 (cit. on p. 44).

Ye, X., & Hong, S. H. (2019). Toward Industry 4.0 components: Insights into and implementation of asset administration shells. *IEEE Industrial Electronics Magazine*, *13*(1), 13–25 (cit. on p. 51).

Yli-Ojanperä, M., Sierla, S., Papakonstantinou, N., & Vyatkin, V. (2019). Adapting an agile manufacturing concept to the reference architecture model Industry 4.0: A survey and case study. *Journal of industrial information integration*, *15*, 147–160 (cit. on p. 56).

Yurin, A., Berman, A., Nikolaychuk, O., & Dorodnykh, N. (2019). Knowledge base engineering for industrial safety expertise: A model-driven development approach specialization. *International Conference on Information Technologies*, 112–124 (cit. on p. 57).

Zachman, J. A. (1987). A framework for information systems architecture. *IBM systems journal*, *26*(3), 276–292 (cit. on p. 63).

ZVEI - Zentralverband Elektrotechnik- und Elektronikindustrie e.V. (2015). *Industrie 4.0: Das Referenzarchitekturmodell Industrie 4.0 (RAMI 4.0)* (tech. rep.). ZVEI. (Cit. on p. 7).

Online sources

Dube, S. (2021, July 19). *Product line engineering and model based systems engineering – overview*. Retrieved August 14, 2022, from https://www.samares-engineering.com/en/2021/07/19/product-line-engineering-and-model-based-systems-engineering-overview/. (Cit. on p. 107)

iED Team. (2019, June 30). *The 4 industrial revolutions*. Retrieved July 28, 2022, from https://ied.eu/project-updates/the-4-industrial-revolutions/. (Cit. on p. 4)

The Open Group. (2022, July 28). *Introduction to the Architecture Development Method (ADM)*. Retrieved July 28, 2022, from https://pubs.opengroup.org/architecture/togaf9-doc/arch/chap18.html. (Cit. on pp. 41, 42, 113)

Tschirner, C. (2015, November 22). *Begriffe des Systems Engineerings Teil 4 - Architekturbeschreibungen – die ISO/IEC/IEEE 42010:2011*. Retrieved April 18, 2021, from https://kem.industrie.de/systems-engineering/architekturbeschreibungen-die-iso-iec-ieee-420102011/. (Cit. on pp. 35, 37)

White, S. K. (2020, June 20). *What is the zachman framework? a matrix for managing enterprise architecture*. Retrieved April 19, 2021, from https://www.cio.com/article/3535909/what-is-the-zachman-framework-a-matrix-for-managing-enterprise-architecture.html. (Cit. on p. 65)

Wikipedia. (2021, April 19). *Zachman framework*. Retrieved April 19, 2021, from https://en.wikipedia.org/wiki/Zachman_Framework. (Cit. on p. 64)