Otto-von-Guericke-Universität Magdeburg

FAKULTÄT FÜR
MASCHINENBAU

Architecture Framework Concept for Definition of System Architectures based on Reference Architectures within the Domain Manufacturing

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieur

(Dr.-Ing.)

von M.Sc. STEPHAN UNVERDORBEN
geboren am 15.01.1990 in Erlangen, Deutschland

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

Gutachter: Prof. Dr.-Ing. habil. Arndt Lüder
Prof. Eugenio Brusa

Promotionskolloquium am 13.12.2021

Preface

This thesis was written during and beyond my employment at Siemens AG - Corporate Technology and in cooperation with the Institute of Ergonomics, Manufacturing Systems and Automation (IAF) of the Chair for Manufacturing Systems and Automation at Otto von Guericke University Magdeburg.

In general, I would like to thank everyone, who in some way or the other encouraged me during the journey of creating this thesis. Especially, I would like to acknowledge involved parties at the Otto von Guericke University Magdeburg, the supervising professors, my colleagues at Siemens AG - Corporate Technology, as well as my family and friends who have supported and advised me during my doctoral studies.

Magdeburg, 25.08.2021 Stephan Unverdorben

Abstract

Increasing complexity as well as continuous changes in the environment of production systems create challenges with regard to the development of systems and their architecture definition. To be able to pursue sustainable business models in the future, the architecture development process should be accompanied methodically and content wise. Especially in connection with the recurring definition of similar systems, reference architectures can make beneficial contributions.

In literature, the use of reference architecture content is not sufficiently addressed in the available architecture frameworks that describe the procedure for developing a complex system. However, if an architecture framework is continuously applied considering a reference architecture, system architectures could be derived from a reference architecture more efficiently and effectively in the long run.

Therefore, the goal of this thesis was to develop a concept for an architecture framework that enables the definition of system architecture descriptions based on a predefined reference architecture description.

In the thesis, as an introduction, the state of the art for production systems engineering, architecture development, and the concept of architecture framework were considered. To verify the identified research gap, relevant architecture frameworks were evaluated with respect to the use of reference architectures. Based on these considerations, in a first step, the "Core Architecture Framework Concept" was developed, which supports the individual development of architecture descriptions during the basic and detailed engineering of production systems. The framework consists of a basic structure and methods for defining content for the step-by-step creation of the architecture description. Based on the "Core Architecture Framework Concept", in a second step, a more advanced method was specified and developed, which takes into account the transition between reference and system architecture descriptions.

These two concepts comprehensively form the "Architecture Framework Concept" for the definition of a system architecture description based on a reference architecture description. Concluding, the content used by the concept proposal was implemented prototypically in the modeling tool MagicDraw and the developed concept was successfully tested and evaluated by using a production system application example.

As a result, it can be summarized that with consistent application of the developed "Architecture Framework Concept" a transition from reference architectures to system architectures is possible, whereby an increase in efficiency and effectiveness during

development of architecture description can be reached. Based on the developed “Architecture Framework Concept”, the operational use should be advanced in further research.

Kurzfassung

Zunehmende Komplexität sowie kontinuierliche und schnelllebige Veränderungen im Umfeld von Produktionssystemen schaffen Herausforderungen hinsichtlich der Entwicklung von Systemen und deren Architekturdefinition. Um in Zukunft insbesondere nachhaltige Geschäftsmodelle verfolgen zu können, sollte der Architekturentwicklungsprozess noch methodischer und inhaltlicher begleitet werden. Gerade im Zusammenhang mit der wiederkehrenden Definition ähnlicher Systeme können Referenzarchitekturen sehr gute Beiträge leisten.

In der Literatur wird die Verwendung von Referenzarchitekturinhalten in den verfügbaren Architekturframeworks, die das Vorgehen bei der Entwicklung komplexer Systeme beschreiben, nicht ausreichend adressiert, obwohl bei durchgängiger Anwendung eines Architekturframeworks Systemarchitekturen effizienter und effektiver aus einer Referenzarchitektur abzuleiten sind.

Ziel dieser Arbeit war es daher, ein Konzept für ein Architekturframework zu entwickeln, das die Definition von Systemarchitekturbeschreibungen auf Basis einer vordefinierten Referenzarchitekturbeschreibung ermöglicht.

In der Arbeit wurde einleitend der Stand der Technik für das Engineering von Produktionssystemen, die Architekturentwicklung und das Konzept des Architekturframeworks betrachtet. Um die erkannte Forschungslücke zu verifizieren, wurden zunächst relevante Architekturframeworks im Hinblick auf die Nutzung von Referenzarchitekturen evaluiert. Darauf aufbauend wurde in einem ersten Schritt das „Core Architecture Framework Concept" entwickelt, das die individuelle Entwicklung von Architekturbeschreibungen bei der Grobplanung von Produktionssystemen unterstützt. Dieses Framework besteht aus einer Grundstruktur und Methoden, mit denen Inhalte für die schrittweise Erstellung der Architekturbeschreibung definiert werden. Basierend auf dem „Core Architecture Framework Concept" wurde in einem zweiten Schritt eine weiterführende Methode spezifiziert und entwickelt, die den Übergang zwischen Referenz- und Systemarchitekturbeschreibung berücksichtigt. Diese beiden Konzepte bilden insgesamt das „Architecture Framework Concept" für die Definition einer Systemarchitekturbeschreibung auf Basis einer Referenzarchitekturbeschreibung. Die Inhalte des definierten Konzeptvorschlags wurden prototypisch im Modellierungswerkzeug MagicDraw umgesetzt und abschließend anhand eines einfachen Anwendungsbeispiels eines Produktionssystems erfolgreich getestet und evaluiert.

Als Ergebnis kann zusammengefasst werden, dass bei konsequenter Anwendung des entwickelten „Architecture Framework Concepts" ein Übergang von Referenz-architekturen zu Systemarchitekturen möglich wird, wodurch ein Mehrwert hinsichtlich Effizienz und Effektivität bei der Architekturerstellung erreicht werden kann. Damit wurde ein Beitrag für die zukünftige Entwicklung von Architecture Frameworks zur weiteren Steigerung der Effizienz beim Bau technischer Anlagen geschaffen.

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 that 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.”

Hausen, 25.08.2021 Stephan Unverdorben

Contents Overview

Table of Contents

List of Figures

List of Tables

Abbreviations

AC	Available content
ANSI	American National Standards Institute
ATAM	Architecture Tradeoff Analysis Method
CAD	Computer-aided design
CHM	Cylinder head manufacturing system
CrESt	Collaborative Embedded Systems
CES	Collaborative embedded system
CPS	Cyber physical system
CPPS	Cyber physical production system
CSG	Collaborative system group
DoDAF	Department of Defense Architecture Framework
DSL	Domain-specific language
EIA	Electronic Industries Alliance
FEAF	Federal Enterprise Architectural Framework
GERAM	Generalized Enterprise Reference Architecture and Methodologies
GL	Granularity layer
GPL	General purpose language
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IIRA	Industrial Internet of Things Reference Architecture
INCOSE	International Council on Systems Engineering
IoT-RA	Internet of Things Reference Architecture
IoT	Internet of things
IPML	Industrial Plant Modeling Language
ISO	Organization for Standardization (German: Internationale Organisation für Normung)
I4.0	Industry 4.0
MBSAP	Model-Based System Architecture Process
MBSE	Model-based systems engineering

MFESA	Method Framework for Engineering System Architectures
MODAF	Ministry of Defense Architecture Framework
NAF	NATO Architecture Framework
NASA	National Aeronautics and Space Administration
OOSEM	Object-Oriented Systems Engineering Methodology
PLC	Programmable logic controller
RA	Reference architecture
RAMI 4.0	Reference Architecture Model Industry 4.0
RC	Required content
RM-ODP	Reference Model of Open Distributed Processing
RQ	Research question
SE	Systems engineering
SoI	System of interest
SPES 2020	Software Platform Embedded Systems
SPES_XT	Software Platform Embedded Systems "XT"
SyA	System architecture
SysML	Systems Modeling Language
SYSMoD	Systems Modelling Toolbox
TOGAF	The Open Group Architecture Framework
UML	Unified Modeling Language
US	United States
VDI	Association of German Engineers (German: Verein Deutscher Ingenieure e.V.)

1 Introduction – Motivation and Problem Context

From its very beginnings the engineering of systems posed complex challenges, which have been met at the time by successful responses to these objections in the form of different practices and principles [1]. The challenges evolved over time due to increasing complexity [2] of the system of interest and its to be considered environment (e.g., because of an increasing number of system functions, components, and their interfaces as well as emerging non-linear interactions between these components [3]). The evolution of the systems engineering discipline is based on these practices and principles and will contribute to the ongoing development of systems engineering and in particular sub-disciplines like model-based systems engineering (MBSE) [1]. Main contributing factors to the current changes experienced in systems engineering can be divided in market-based changes and technological developments. Market-based changes include rising globalization [4], an increased number of competitors [5] and competitive market pressure [6], changed customer needs for high quality but individual systems [4, 7–9], and a continuous demographic change [4]. On the other hand, technological developments and trends such as Industry 4.0 (I4.0), cyber-physical systems (CPS), Internet of Things (IoT) [1], and, among others, related specific requests for adjustment of lot sizes (up to lot size 1) as well as the mass customization of products [10] trigger these changes. Ultimately, these changes will lead to an increasing complexity of the considered systems and place additional requirements on the design and operation of these systems [2, 3, 11–13]. In addition to the increasing complexity, constant changes in the environment of a system, shorter life cycles, innovations with respect to processes and materials, advancing globalization and possibly distributed engineering efforts will lead to uncertainty, demands on both the development process and management of a system with respect to cost, quality and time, and on integration as well as control of the different life cycles, disciplines, stakeholders, projects, specific artifacts, and tools [1, 5, 14–19]. Furthermore, challenges arise with respect to the topics of knowledge management and the reuse of existing solutions, which result from the dynamics and short life time of the system and its environment [17, 20], as well as from the complexity of work tasks and in part the requirements for the qualifications of the affected stakeholders and their workplaces [4]. The direction systems engineering is gravitating towards in the future is shaped by the global environment and its social and human needs, the related global trends, and the available technologies systems are realized upon [3]. Based on these effects the work environment in which systems engineering is practiced will change [3]. Environmental change and sustainability,

interdependent economies, and increasing globalization will more than ever foster the fact that the creation of systems has to be considered with respect to the global environment and its needs [3]. Over time standards and handbooks have emerged to channel knowledge and provide a basis for systems engineering within the different domains [3]. Depending on the application domain the discipline of systems engineering is applied differently [3] and is mostly dealing with domain depended challenges. When focusing on the manufacturing domain level and considering the engineering of production systems, main challenges are:

- Alignment and evolvement of systems engineering with increasing system complexity [3].
- Establishment of a theoretical systems engineering foundation and structured procedure across different system types, industries, and organizations [3, 21].
- Enabling integration across involved disciplines and specialists, between development phases and engineering projects [3, 21].

In conclusion can be said that these changes affect a wide range of areas involved with systems engineering and pose challenges to systems, stakeholders, processes or organizations.

Challenges not taken into account sufficiently will, among others, result in uncertainty or negative effects on the system and its performance, which might lead to necessary changes as well as additional expenses [15]. If those challenges are not addressed, for the most parts, the result will be a loss in reputation [18], and the goals set in development projects, such as time, costs and quality, will increasingly not be achieved to the planned extent [22]. Therefore, future engineering is forced to implement systems withstanding those challenges, which will require, among others, focusing on the implementation of systems meeting stakeholder needs (such as adaptability), integration of relevant stakeholders in business and value creation processes, efficient utilization of resources, and (ergonomic) design of systems and processes [10, 23, 24]. To comprehensively manage current and future challenges, approaches such as systems engineering and related disciplines like model-based systems engineering are needed [22], which focus on interdisciplinary and system comprehensive methods for the development of systems [16]. If this necessary shift is not processed seriously, the differences between established discipline-focused development methods and the required development methods will worsen due to the increasing complexity and make it much more difficult to engineer a system in the future [16].

According to [3] by solving existing challenges, future system engineering will

- have a broader scope of application domains in order to fulfill the growing demand of sustainable and globally competitive system solutions.
- more comprehensively involve market as well as environmental and social demands in consideration for system life cycle decisions and long-term risks.
- have a main role in integrating all relevant disciplines across different organizations, projects, and regions.
- provide a more comprehensive theoretical foundation as well as model-based methods and tools, which will allow the relevant stakeholders involved in the systems engineering process to better understand increasing complexity and to make more comprehensive decisions in uncertain system environments.
- be carried out by a wide range of professionals over different domains supported by an extensive knowledge infrastructure, methods, and toolsets.

To transform the current status quo of systems engineering, the main challenges need to be assessed, possible solution approaches have to be defined, and implemented successfully [3]. This path must be followed jointly by different groups such as industry, government, and research in order to create a broader knowledge base and general understanding and recognition among the relevant stakeholders [3]. This is the necessary basis for successfully establishing competitive system engineering frameworks, model-based procedures, and tools [3]. These capabilities and knowledge will enable the ability to tackle newly occurring research challenges in a successful manner in the future [3].

The development and implementation of production systems requires the creation and use of system structures and documentation of relationships of elements of this structure. The latter demands engineering processes that are distributed across different disciplines as well as stakeholders and whose application becomes more difficult and complicated with increasing system complexity. While developing a system, different phases are run through, ranging from system design and creation of the architecture to domain-specific design and system integration [25]. The architecture and its description are an integral part of the engineering process and of importance, since these determine fundamental decisions in the early phases of the life cycle of a system, which are influencing the remainder of the system's life cycle. Thus, the total life cycle costs of the system to be developed are affected in a great measure [18, 26]. An architecture may serve several purposes, such as re-using system elements, context definition, design guidance, and complexity handling [27]. In

general, different architecture types can be distinguished, for example, system architectures (SyA) and reference architectures (RA). Reference architectures can be used as a template for the definition of specific systems and can, among other things, reduce development risks and efforts, and increase effectiveness, quality, and reuse [28, 29]. The use of a reference architecture is creating positive impact and potential (competitive) advantages, among others, "[...] in environments with a high multiplicity factor, creating social, organizational, business, application and technical complexity" [29]. Most of the time, the application of a reference architecture is appropriate if many specific architectures for systems of a group/domain are to be continuously created and thus savings can be made cumulatively over several projects [28]. For the definition of both the reference architecture and the system architecture an architecture framework can be used. An architecture framework offers a basic structure in combination with associated methods and tools and describes a procedure how an architecture can be implemented and sustained in a certain domain [30, 31]. By using a reference architecture to define system architectures, the speed to market can be improved, the market price can be kept competitive, subcontracting and procurement can be made more efficient due to the standardized structure, expenses for integration and test can be reduced, and training, field support and maintenance can be improved [28]. To be able to achieve these advantages within architecture development existing changes and challenges in systems engineering must be addressed [3]. "For an efficient implementation of systems engineering not only the procedure itself is relevant but also key enablers like capable and effective tools, people, and processes" [2]. However, when looking at the relevant literature, it is noticeable that the terms coined are widely used and considered useful but are not sharply separated from each other. The existing approaches are mostly limited to general descriptions and well-known frameworks often focus on more abstract enterprise level architectures. On a specific level, it shows that there are hardly any concrete approaches considering the use of reference architectures for the derivation of system architecture descriptions in the area of discrete manufacturing. Based on that situation the following problem statement arises for this thesis.

Problem Statement

For the architectural considerations during the engineering of production systems, the architecture frameworks currently available do not satisfactorily describe the derivation of system architecture descriptions utilizing a suitable reference architecture description.

1.1 Research Objective and Scope

In relation to the challenges and achievable benefits described in the previous section, a research hypothesis is formulated that describes the goal and focus of the thesis.

Research Hypothesis

By consistently applying a suitable architecture framework for production systems, system architectures can be derived from a reference architecture efficiently and effectively.

In order to achieve this goal and to better position the architecture process in relation to the formulated challenges, the thesis deals specifically with the specification of such an architecture framework and how it can be used for the creation of discrete production system architectures based on a reference architecture. The stakeholders as a source of knowledge and creativity within an architecture process cannot and should not be replaced by a framework but support their work. In addition, it should be noted that a more holistic view of the entire organization concerned is likely to deliver better economic and strategic results in the long term. However, in the context of this thesis a more detailed scope is deliberately placed on the architectural considerations during systems engineering and the transition from reference to system architecture to therefore be able to develop a solution concept for this specific aspect and present it for discussion. If proven, the concept can be integrated into a more comprehensive architecture framework with a broader scope at a later point of time. Based on these criteria, the following research objective emerges for the thesis.

Research Objective

The main research goal is to develop an architecture framework that can be used to derive system architecture descriptions for production systems from predefined reference architecture descriptions.

1.2 Thesis Outline

Based on the defined problem as well as the research hypothesis and objective, the following outline results for this thesis.

Chapter 1 examines the initial situation and frames the problem statement. Also, initial input for the following chapters was presented.

In chapter 2 and 3 the current state of the art is considered, serving as a starting point for all further specifications and definitions made within this thesis. **Chapter 2** implies the definition of relevant terms and serves as introduction into the world of systems engineering as well as into the subdiscipline of the architecture development (chapter 3). Chapter 2 commences in defining the system concept, the system environment, and the system life cycle as the central starting point for the topic of systems engineering, which is subsequently described. A possible process for the development of such a system, the consideration of possible effects of systems engineering on the life cycle and the total costs of a system as well as the subdiscipline of model-based systems engineering conclude chapter 2.

In **chapter 3**, this is followed by a general discussion of the architecture topic, the content and purpose of architectures, and a possible creation process. The two architecture types, system architecture and reference architecture, and their relationships to each other are described in more detail afterwards. Finally, the concept of architecture frameworks is discussed as well as the existence of available frameworks, which might be applicable for the utilization of reference architecture content to define system architectures, examined.

Based on the state of the art, **chapter 4** processes the research objective into more detailed research questions.

Chapter 5 followingly deals with the specification of the architecture framework structure, the definition methodology of architecture content, as well as predefined architecture content types on the one hand. On the other hand, the methodology for the consideration of reference architecture content for the specification of specific system architecture descriptions is examined.

The prototypical implementation of the framework in the form of a domain-specific language in a modeling tool as a model-based systems engineering approach is described in **chapter 6**. For evaluation and based on the findings of chapters 5 and 6, a prototypical application of the architecture framework for the derivation of a system architecture based on a reference architecture is presented in **chapter 7** whilst discussing the results and relating them to the specified architecture framework.

Finally, **chapter 8** concludes all findings with respect to the defined research questions and potential objectives for future research are named. An overview of the above defined structure and the described chapters are shown in Figure 1.

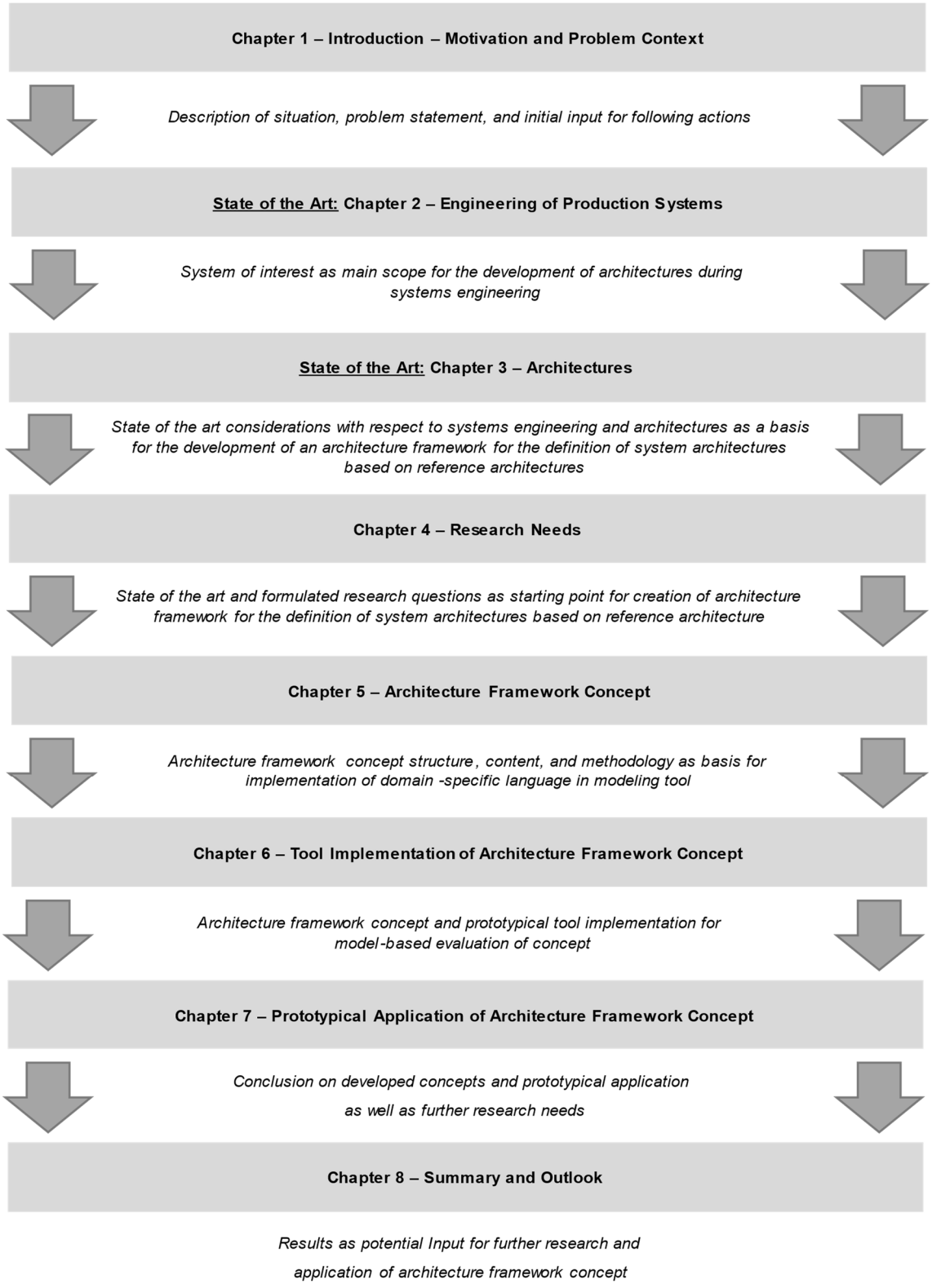


Figure 1: Thesis outline

1.3 Link to Research Project Collaborative Embedded Systems

The author of this thesis contributed as a part of Siemens AG - Corporate Technology to the research project CrESt- Collaborative Embedded Systems, which was funded by the German Federal Ministry of Education and Research, and that some of the knowledge gained during the work on the project was incorporated into this thesis. The research project CrESt, which had a duration of over 3 years, involved 23 partners from academia and industry with the goal of defining new development methodologies and techniques for collaborative embedded systems [32]. As a starting point of the project CrESt, the methods and tools created in the two predecessor projects "Software Platform Embedded Systems" (SPES 2020 [33]) and "Software Platform Embedded Systems_XT" (SPES_XT [34]) were utilized [32]. Those methods and tools have been developed to handle development process complexity to support the continuous model-based development of embedded systems [32].

The focus of the author's contribution was mainly on the contents of the first engineering challenge EC1 - flexible architectures and the utilized application scenario with respect to the adaptable and flexible factory. The structure of the project and its main research topics are shown in Figure 2. Results of the research project have been published in the book "Model-Based Engineering of Collaborative Embedded Systems - Extensions of the SPES Methodology" [35]. For the evaluation of the specified architecture framework one of the project demonstrators as an example for a specific system (An architecture viewpoint describes a “work product establishing the conventions for the construction, interpretation and use of architecture views to frame specific system concerns.” [30], see section 1.4) as well as a in the book specified reference architecture is used (see section 7.2).

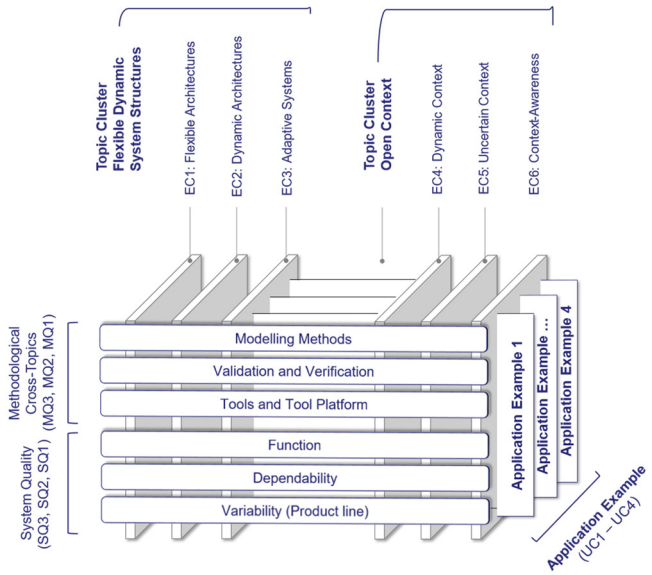


Figure 2: Structure of research project CrESt (adapted from [32])

1.4 Production System Application Example

As a continuous application example throughout this thesis a demonstrator applied within the previously described research project CrESt will be used to practically explain different theoretical approaches and methods. Subsequently, the model is used for the concluding evaluation of the specified architecture framework concept. A CAD-model (computer-aided design) as well as the real-life implementation of the production system application example are shown in Figure 3 and in Figure 4.

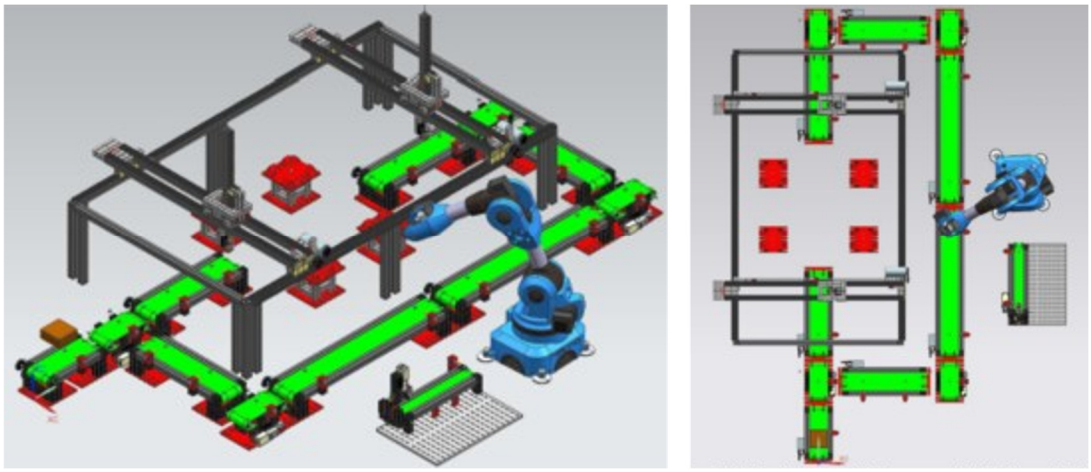


Figure 3: CAD-model of production system application example [36]

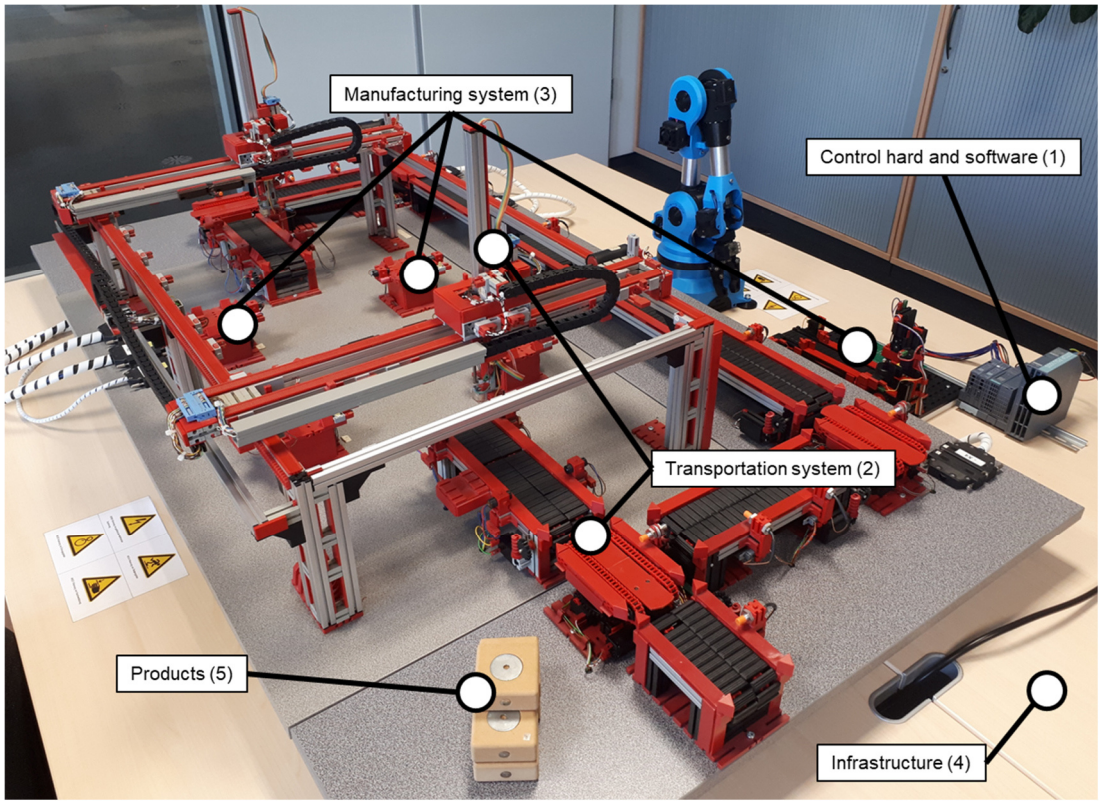


Figure 4: Production system application example (adapted from [36])

The goal of using a simplified model of a production system in the context of this thesis is to represent complex and/or abstract concepts and relationships of a system, with the purpose to convey them in a comprehensible and understandable way. The utilized example embodies a simplified model of a production system for the discrete manufacturing of cylinder heads. A cylinder head sits on an engine block and is an integral part of the overall engine structure, for example, in passenger cars. In general, a product is, as the "result of a process" [37], created by a "set of interrelated or interacting activities that transforms inputs into outputs"[37]. In case of the production system application example, the process of manufacturing and the cylinder head, both are described in more detail in the following. The product "cylinder head" and its production process are shown in Figure 5.

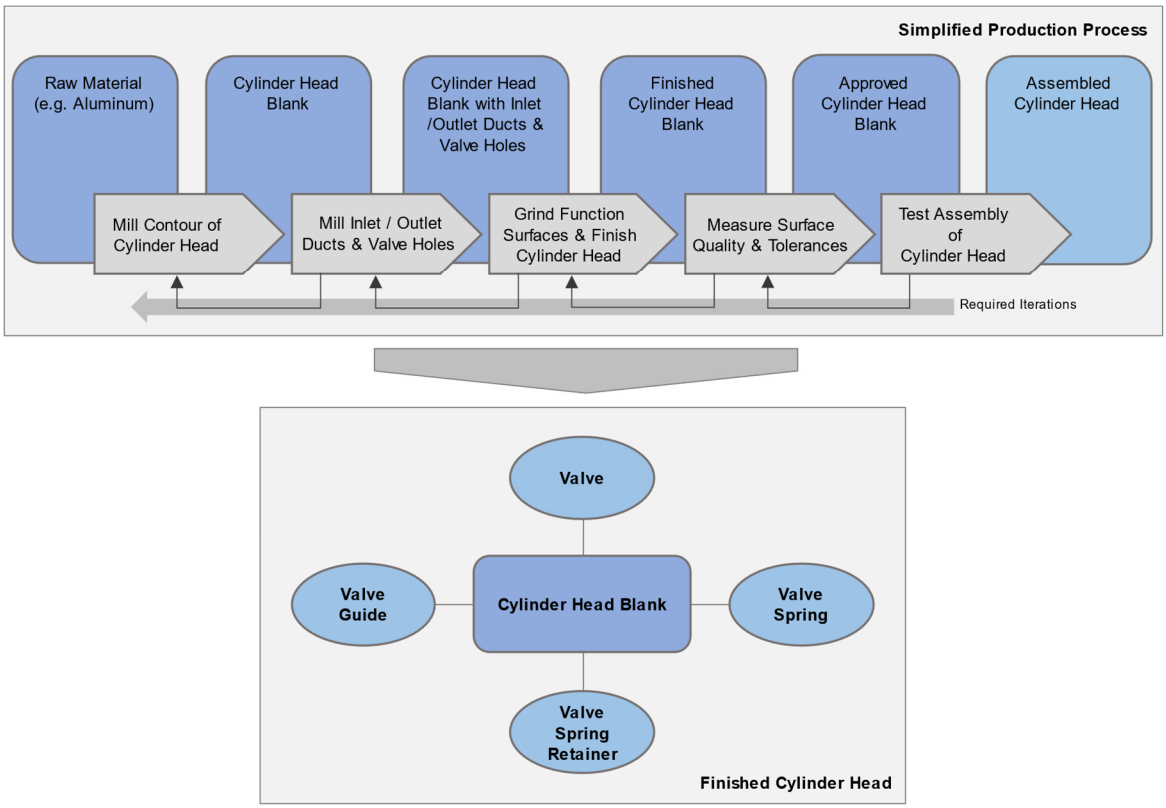


Figure 5: Simplified representation of cylinder head manufacturing process

The manufacturing of the cylinder head is divided into five work steps, which are mimicked by the production system application example. In a first step, starting with the basic shaping of a cylinder head, the contour of the cylinder head is milled from raw material (e.g., aluminum). In a second step, the corresponding inlet and outlet ducts and the valve holes are milled/drilled into the cylinder head blank. After shaping is complete, functional surfaces on the cylinder head are machined (e.g., by grinding) and the entire cylinder head is finished (e.g., by polishing). After machining, the

cylinder head is inspected for specified surface quality, dimensional accuracy, and tolerances. In the event of an unsatisfactory result, a decision must be made whether the product can be revised or must be declared as scrap material. This is assessed after each work step. After successfully checking the quality, in the test assembly step, the valve guides will be fitted to the milled valve holes and the valves and valves springs will be installed and held in place by the valve spring retainers. The result of this manufacturing process is represented by the finished and test wise assembled cylinder head. Further steps such as engine construction are not considered in this example. In addition, it shall be assumed that the cylinder head will not be cast but milled and that it only holds valves, valve guides, valve springs, and valve spring retainers. It is agreed on that the production steps are sufficient for the manufacturing of a cylinder head. In order to produce a cylinder head, an adequate production system for its manufacturing as well as related stakeholders[1] and resources[2] are needed. With respect to the manufacturing of cylinder heads such stakeholders are, for example, the operator of the system, or engineers involved with health & safety or the maintenance of the system. The production system itself is shown in Figure 4 and represents different components utilized for the manufacturing process of the cylinder heads. The production system application example can be divided into the control related hard and software (1), transportation systems (2), manufacturing systems (3), and infrastructure (4). The cylinder heads are represented by cubes (5). The production system itself is controlled by the operator via a corresponding interface and associated hard and software. The necessary signals are forwarded to a PLC (programmable logic controller) that is connected to the model and processes the available input and output signals. The five processing stations for the production of cylinder heads are connected by different transport systems (e.g., gantry crane, conveyor belt, and robot). The system connects to and utilizes the available physical infrastructure. After the manufacturing process has been triggered by the operator the resources, later semifinished, and finished products, represented by coded blocks, are transported to and between the different processing stations. in the example Those stations emulate the process required for the production of cylinder heads, which is shown in Figure 5.

[1] The term stakeholder describes an individual person or group of persons who have one or more concerns about the system of interest and an interest in the system meeting the needs and goals of that person or group [38, 39].

[2] Resources describe an "asset that is utilized or consumed during the execution of a process" [8]. The resources include, for example, “funding, personnel, facilities, capital equipment, tools, and utilities such as power, water, fuel and communication infrastructures” [37].

In addition, information on the utilized production system application example can be found in [36] and [40].

The presented example will be used throughout the thesis and explanations related to the model described above are symbolized by double-framed boxes. Such examples can be found, among others, in Figure 10 and Figure 24.

Example

Descriptive explanation of thesis content with respect to the introduced production system application example.

2 Engineering of Production Systems

Within this chapter the discipline and objectives of engineering, the system topic itself, as well as related terms and relations will be introduced and specified to create a foundation for the introduction of the developed architecture framework. As this thesis is focused on the definition of an architecture framework mainly considering the specification of system architectures based on reference architectures, in section 2.1 the term system and directly related terms will be defined. Section 2.2 will provide an overview of the typical life cycle of a system and will lead into the topic of systems engineering (see section 2.3). The state of the art with respect to the topics architecture framework, system as well as reference architecture will be introduced in chapter 3. Figure 6 gives an overview on the content of the following section.

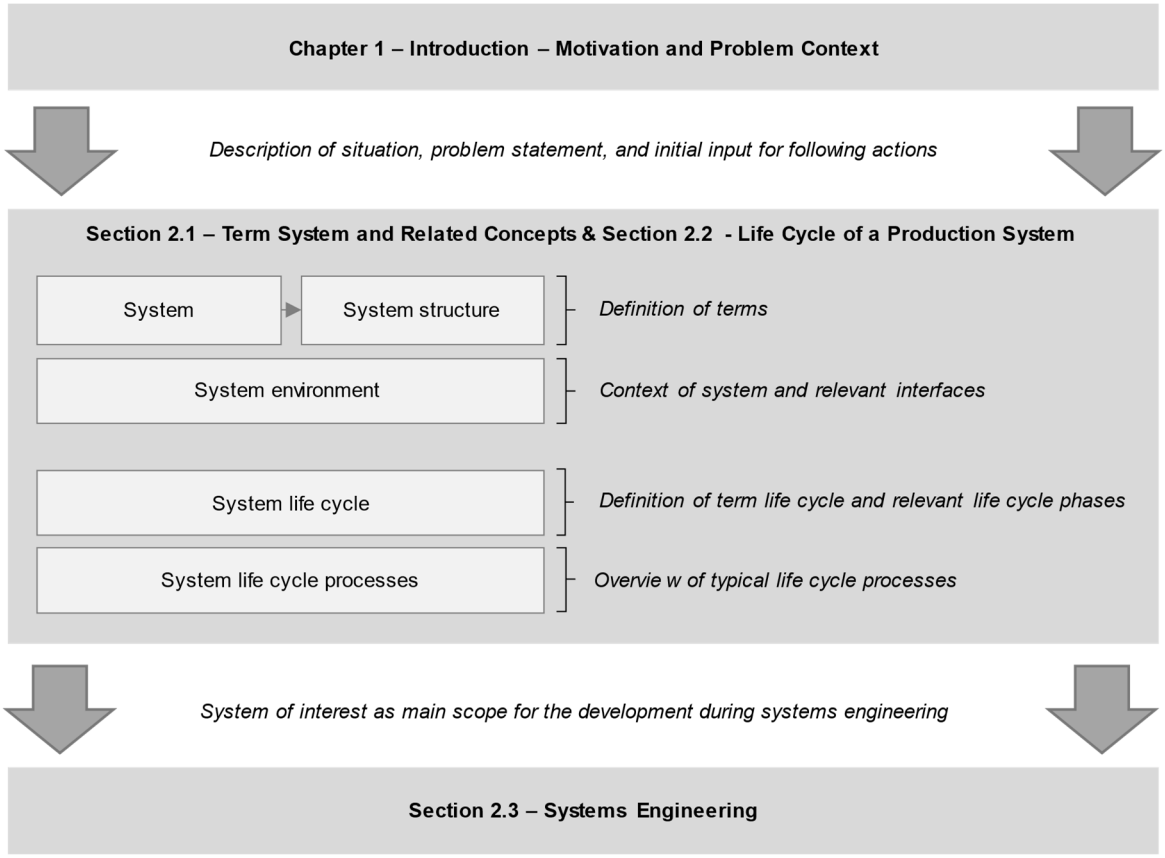


Figure 6: Overview of content of section 2.1 and section 2.2

2.1 Term System and Related Concepts

As the term system represents a reoccurring term in this thesis and is often understood differently within research and society, in the following, definitions from different authors are examined and a concluding definition of the term system will be given. In

addition, the structure of a system and its surroundings will be discussed. The goal of describing a system, its internal structure and related surroundings aids the purpose to create a clear definition of the system concept and its use in the context of this work.

2.1.1 Definition of Term System

As the characterizations of the term system are manifold, the full definitions considered in the following are given in the annex in Table 8. Those definitions are mainly based on sources with relations to the topic of systems engineering, which is the core field of consideration within this research. Based on the most important aspects of the applied definitions and their representation within the available definitions it is determined if an enhanced definition is needed or if an existing definition can be used within this thesis.

When analyzing the introduced definitions, first and foremost it can be stated that the contents of the individual definitions do not contradict or mutually exclude each other. In fact, the analysis shows that two key statements can be found in almost all definitions. Dependent on the definition those two statements are detailed by more specific descriptions.

(1) *A system consists of an integrated, interoperable and interacting set of elements* [26, 37, 38, 41–47]. A single element is part of the set of elements, which constitutes a system [37]. Those elements represent, for example, equipment, facilities, personnel, and services [26, 37, 42–44]. The term “integrated” describes a considered system being assembled out of different elements (e.g., subsystems or assemblies [26]), which form a certain structure [38, 46]. These elements within the system must fit together (interoperable) in order to achieve the stated purpose of the system [46]. By interaction between the elements a system adds value, which cannot be achieved by a single element [43, 44].

(2) *A system achieves a specified objective /need* [26, 37, 38, 42, 43, 45, 46]. The system provides products and services in order to achieve an objective / need specified by users and stakeholders [30, 38].

The above specified statements can be considered as the bare minimum for a definition of a system. In addition to those two main statements the different definitions highlight specific other sub-facts. One fact, which is mentioned repeatedly is the systems environment.

(3) *A system operates in a dedicated environment* [30, 37, 46]. This suggests that every system is defined with a concrete application in mind and in order to achieve its objectives in that specified environment.

As a summary it can be stated the definitions given in Table 8 provide several common main points with respect to the term “system”. Some of the definitions nearly include all contents, but do not capture all aspects in detail. Therefore, based on the common aspects a definition of the term “system”, which will be used within this thesis, will be given below.

System

A system consists of a fitting set of interacting elements which form a certain structure and are composed to achieve a stated purpose and goal within a defined specific environment.

In order to put the terms within the definition into more perspective, system of interest, system structure, and system environment will be described below. Their dependencies among each other are summarized in Figure 7 to Figure 9.

System of Interest (SoI)

A system of interest describes a system which is in the focus of consideration within e.g., a design/engineering process. During the engineering phase the term system of interest (SoI) refers to the considered system, which has to be developed. In literature the term is defined as follows.

System of Interest

The system of interest describes one specific system and its life cycle as well as architecture considered during the process of preparing a description of the architecture of the system [30, 37].

In terms of system architecture and reference architecture considerations it is assumed that there is only one system of interest being regarded. For the reference architecture it is further assumed that the system of interest represents the

requirements and characteristics of all systems in the considered group of systems (see sections 3.2 und 3.3).

The definition of a system of interest allows a better distinction between components of the system and their system structure from the surrounding system environment. The distinction between the terms and a definition is given in the following paragraphs.

2.1.2 System Structure

As specified within the definition of the term system, the elements of a system form a certain structure to achieve a stated purpose. That hierarchical classification of system elements is done and worded differently with its individual downsides and benefits within the different definitions; for example, elements, subsystems, or assemblies [26]. In order to obtain a consistent wording within this thesis, the concept defined in ISO/IEC/IEEE 15288 [37] will be shortly described below and used within the following chapters.

The relations between a related set of interacting elements and the considered system can be displayed in their hierarchy [37]. Within the structure, the elements are classified into different levels [41]. Each system has zero or more subordinate system elements, but every system element has at least one superordinate system (see Figure 7) [41]. The subordinated system elements represent architectural elements of a larger system [38].

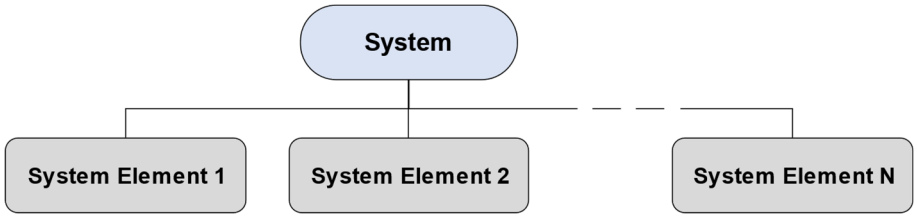


Figure 7: General structure of a system (adapted from [37])

For a holistic representation of a more complex system of interest, on a subordinate hierarchy level, a system element can also represent a system with its own subordinated system elements [37]. This hierarchical concept can be utilized on the different hierarchical level in order to fully represent the considered system of interest as shown in Figure 8.

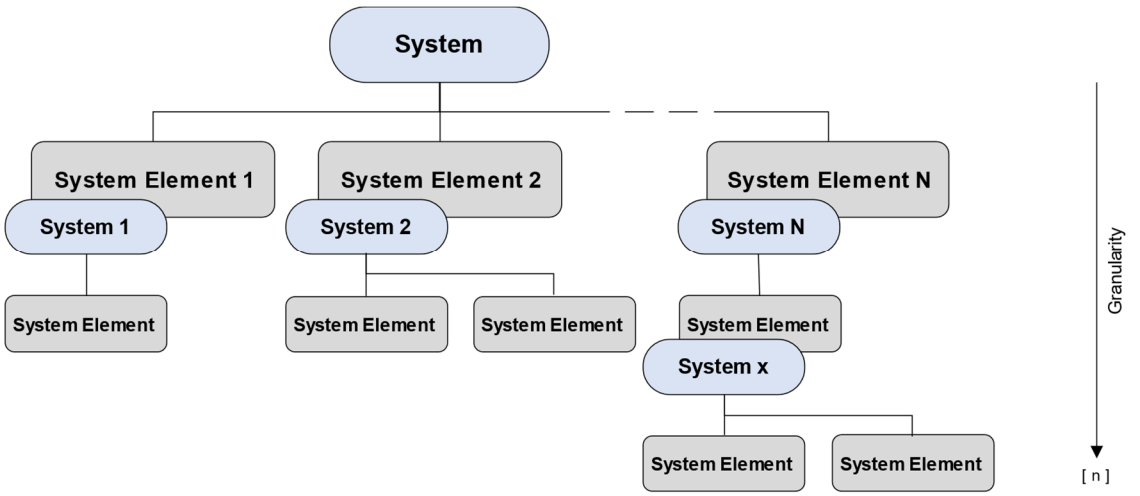


Figure 8: Representation of system of interest structure (adapted from [37])

The hierarchy within a system that is created by structuring is divided into several levels, as shown in Figure 8. The representation of the system is thus detailed through the different levels, starting at the top with the lowest level of detail to the bottom with the highest level of detail. These levels are referred to as levels of granularity. Granularity describes the condition, e.g. of a system, to be composed of many interrelated elements [48]. The distinction of different levels of granularity contributes to an easier decomposition of a system into its structurally related elements. This allows, for example, the decomposition of complex systems into smaller, less complex structures, which in case of systems engineering is a key concept [49]. In summary, the term granularity can be defined and used as follows.

Granularity

"The level of detail considered in a model or decision-making process. The greater the granularity, the deeper the level of detail. Granularity is usually used to characterize the scale or level of detail in a set of data" [50].

2.1.3 System Environment

Depending on the literature, different terms are used synonymously to describe elements outside the system boundaries. Since most of the literature shares the same ideas but uses different terminology, a clear allocation of definitions to terms describing the outsides of the system boundaries is necessary. Therefore, the terms environment, (system) context, and surroundings are introduced and described.

The term environment is often used synonymously with the term system context, e.g. in [30]. However, this assignment is not quite sufficient as, from the authors point of

view, the two terms describe different contents. Therefore, within this thesis the environment of a system includes all elements outside the system boundary. This implies, that elements which are not relevant for the definition of the system are included as well. This concludes in the environment representing the entirety of context (relevant content) and surroundings (irrelevant remaining content). Environment is regarded as follows.

Environment

The environment of a system considers all elements outside the defined boundary of the system of interest. Therefore, the environment includes relevant and irrelevant elements with respect to the definition of the system of interest. Elements of the environment might be connected by relationships among each other and to the system of interest.

The context of a considered system represents elements which do not belong to the system itself, but do have a relationship and interactions with the system [26]. These context elements can represent, for example, systems, stakeholders, and environmental effects. The context and the system of interest are separated by a system boundary [26] and connected by their touchpoints or in the following named as interfaces. The system boundary defines which elements are considered as the system of interest [26]. For the later consideration of reference and system architectures, it should be noted, that the defined context boundary and thus the indirect determination of the content of the system contributes to the determination of the abstraction level of the system of interest (see section 0). The term (system) context is defined as follows.

(System) Context

The (system) context is "[…] determining the setting and circumstances of all influences upon a system" [30].

As mentioned above within the environment of the system, a differentiation between the relevant context and irrelevant surroundings has to be made. This is important in order to fully describe the context of the system of interest. The context and surroundings are separated by the context boundary. The term surroundings is defined below.

Surroundings

The surroundings describe the elements within a system environment, which are not relevant for the definition of the system and do not have any relationship to the system of interest.

The clear definition of boundaries and interactions is very important for the later engineering of a system. In order to clearly distinguish what needs to be considered to obtain a suitable system design which can fulfill the planned purposes/goals and provide benefits for its stakeholders, the mutual dependencies have to be known. The terms, concepts, and their relationships as introduced above are depicted in Figure 9. The graphic shows the system of interest with its defined structure, as well as the shifting interface boundaries (dashed line) between system of interest, context, and surroundings. The boundaries change according to their dependency on the application example considered.

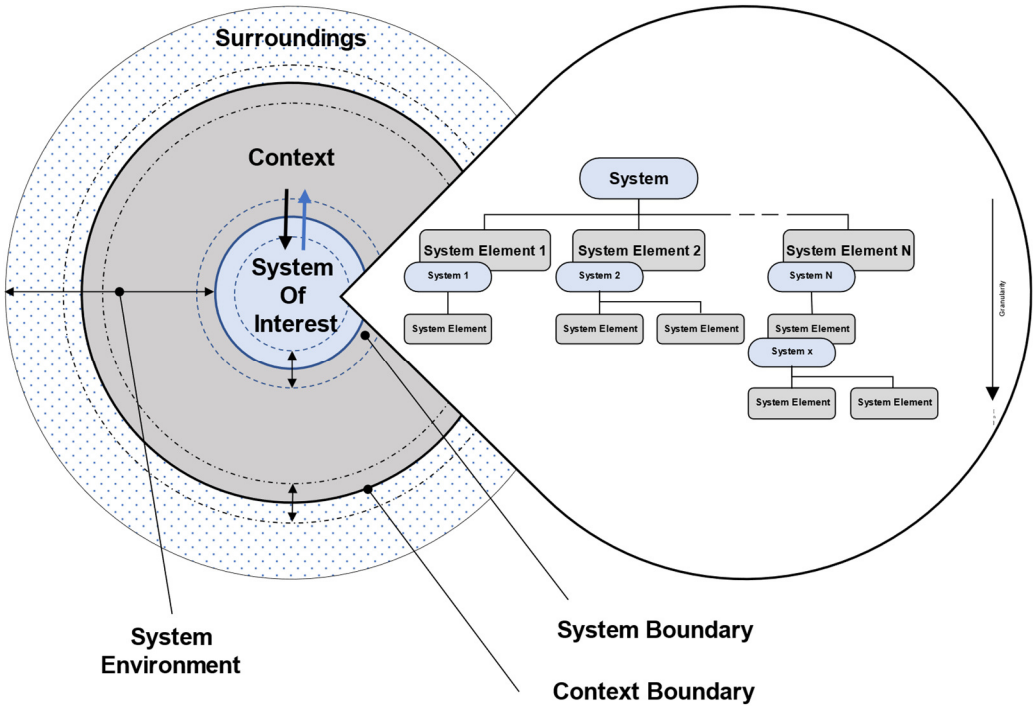


Figure 9: Dependencies between system and context - adapted from [26, 37, 46]

Production System Application Example

With regard to the concepts "system", "system structure", and "system environment", it can be stated that the cylinder head manufacturing system represents the system of interest in this thesis. As described, the system consists of different elements, which themselves embody a system on a more detailed granularity layer. In concrete terms, this implies that the cylinder head manufacturing system consists of elements such as "control related hard and software", "transportation system", "manufacturing system", and "infrastructure". On a more detailed granularity layer the element "manufacturing system" represents an independent system, which in turn again consist of elements like milling system, grinding system, quality assurance system, and assembly system.

The system of interest then delimits itself to its relevant context and defines relevant points of contact between itself and elements of the context. The context includes elements like storage or the order processing. This means, for example, that the actual handling of order processes is not part of the system of interest but that it influences the system of interest in the form of the required manufacturing process and the product needed to fulfill specified orders. Those influences shall be taken into account in the form of, for example, requirements. Surroundings include the elements existing in the environment of the system of interest, but the latter have no influence on the cylinder head production and can thus be neglected. These elements include, for example, other production lines in the plant or the workers' social area.

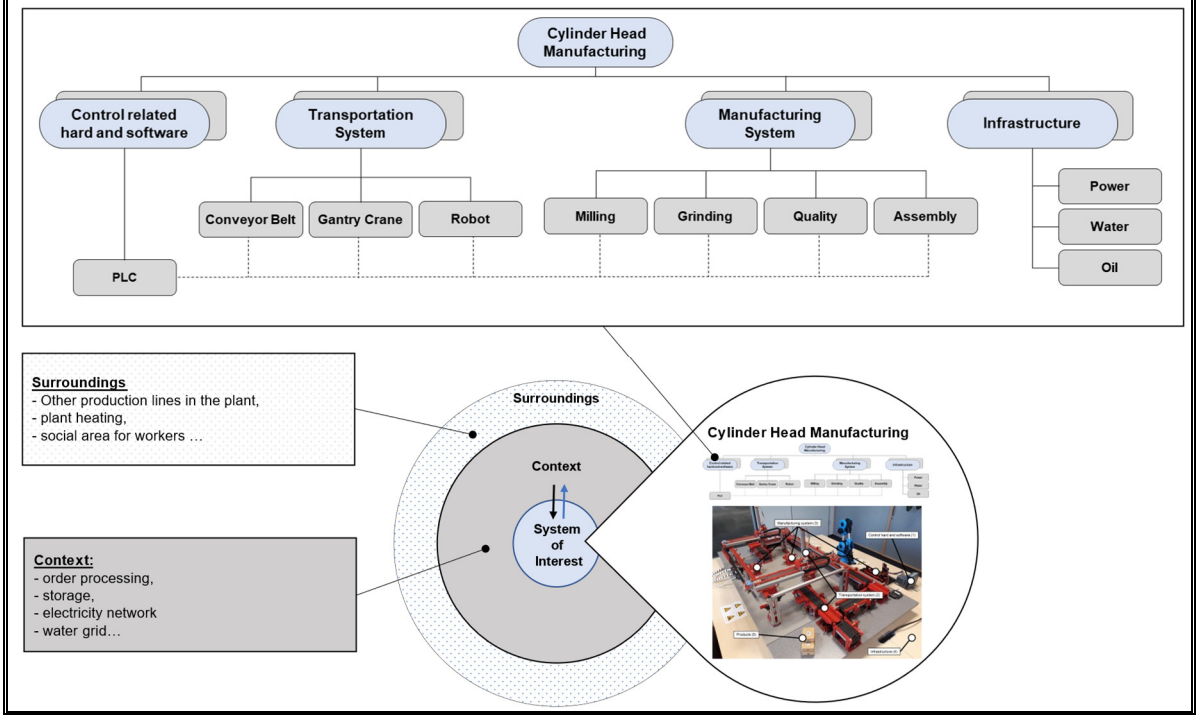


Figure 10: Production system application example – system structure and environment

2.2 Life Cycle of a Production System

After introducing the term system, the general structure, and relationships within the system and between the contained elements, the goal of this section is to introduce the different stages and relationships a system is going through in its lifetime. In addition, considering the focus of this thesis, the relevant processes during the technical focus of the engineering of a system are presented. The purpose of the section is to develop a better understanding of the system and its life cycle phases as well as the early development phases of a system, as these outline the relevant content for the architecture framework presented later.

2.2.1 System Life Cycle Definition

In Fairlay, Forsberg et al. [51] and within ISO/IEC/IEEE 24748-1:2018 [52] is argued, that despite the different variants of system origins, all systems go through a basic essential set of characteristic life cycle phases during their existence. Generally speaking, each system exists for a defined period of time. This period of time can be longer or shorter depending on the system of interest. The period begins with the first appearance of the system and ends with its dissolution. The time span is called system life cycle. In the following, general definitions of the terms life cycle and life cycle model are given.

Life Cycle

A life cycle defines an “evolution of a system, product, service, project or other human-made entity from conception through retirement” [37].

Life Cycle Model

A life cycle model is a “framework of processes and activities concerned with the life cycle that may be organized into stages, which also acts as a common reference for communication and understanding” [37].

In literature different concepts are used to describe the life cycle of a system. This is mainly due to the nature of the system, its use and purpose for which the life cycle concept was derived [52]. A detailed overview and further documentation of different life cycle concepts, e.g. the NASA Project Life Cycle [43] and US Department of Defense Acquisition Process [42, 53], is shown in [26] and was derived from [54]. With respect to the topic of this thesis and according to ISO/IEC/IEEE standards 42010 [30],

12207 [55], and 15288 [37] there are also a life cycle and processes for architectural design with respect to software and systems engineering. The well-known INCOSE Systems Engineering Handbook [26] also utilizes the life cycle of ISO/IEC/IEEE 15288-2015. Both ISO/IEC/IEEE 12207-2017 and ISO/IEC/IEEE 15288-2015 refer to the life cycle and associated life cycle model of ISO/IEC/IEEE 24748-1:2018 [52]. Therefore, and with respect to the considered topics in this thesis the life cycle concept of the ISO/IEC /IEEE 24748-1:2018 standard, which is recognized and widely used in science and industry, will be used as the basis for all further considerations regarding the system life cycle.

In accordance to ISO/IEC/IEEE 24748-1:2018, systems pass through various stages during their lifetime in which they are conceptualized, developed, produced, utilized, supported, and finally retired [52]. This transition is depicted in Figure 11, which represents the life cycle model and the six life cycle stages of a system of interest. The interfaces between the dependent and overlapping individual stages represent the points at which it is decided whether or not the system is in a state to transfer to a subsequent stage [52]. The use of a life cycle model, the associated testing, and step-by-step progression through the stages reduce the risk of incomplete progress and the emergence of inconsistencies while simultaneously mapping the actual progress, which can help relevant stakeholders to maintain an overview in large frameworks and to make important economic decisions [52]. It should also be noted that the uniform presentation of the individual stages in Figure 11 does not indicate the actual effort and time required to complete a stage [52]. The system transfers through the individual stages triggered by different actions in its individual stages, which are committed by stakeholders within the respective phases [52].

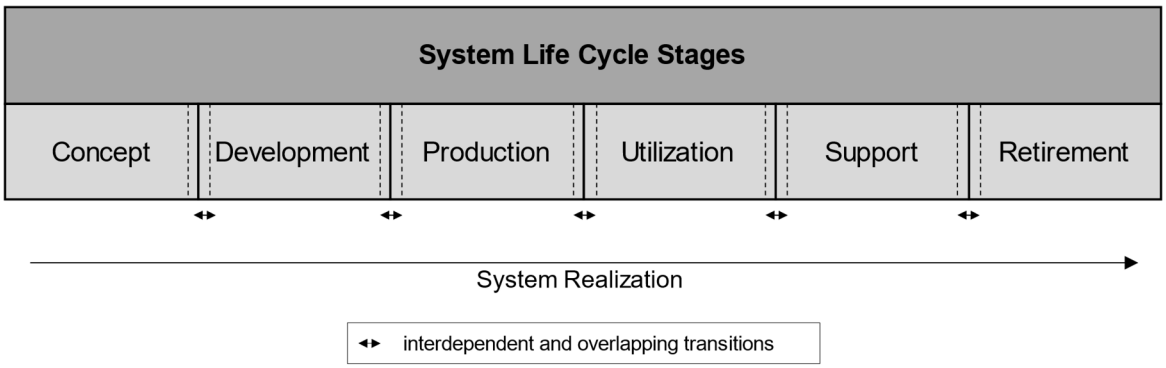


Figure 11: System life cycle model adapted from ISO/IEC/IEEE 24748-1:2018 [52]

Concept

The concept stage begins with the identification of an initial need or requirement. To satisfy the discovered need/requirement, various options are developed and analyzed. Ideally, initial requirements and feasibility studies for the system of interest are carried out in the concept phase. On the basis of these results, it is decided whether a development of the system is pursued further. The purpose of the concept stage is to examine different business models and concepts, find stakeholder needs, propose possible solutions, and define initial requirements. [52]

Development

In the development phase the requirements for the system of interest and all related elements are detailed and specified. During this detailing process, constraints on the design of the system from stakeholder of other stages are also collected and included. Finally, feedback is gathered from relevant stakeholders, e.g., those who produce or operate the system. The result is a final system of interest with all relevant documentation. The purpose of the development stage is to define the system of interest, to refine and finalize the requirements for the system, to create a solution description, to build the system prototypically, and to verify and validate it. [52]

Production

In the production stage, the final concept for the system to be produced is approved and then individually manufactured, assembled, incorporated into the existing environment, and tested. The production stage can extend over the rest of the system life cycle due to e.g., improvement measures or redesign. During the production stage the purpose is to manufacture, inspect, and test the designed system. [52]

Utilization

The utilization stage begins after the system is commissioned and describes the period of time during which the system is used in its intended environment to provide products and services to meet existing needs. In this stage, the focus is primarily on continuous and cost-efficient operation. The stage ends with the transition of the system into decommissioning. During the utilization stage, products and services can evolve, which can then impact the system and its environment. The purpose of the utilization stage is to operate the system to produce the desired product, to offer a service to fulfill the customers' needs, or to foster effectiveness of operations. [52]

Support

On the one hand, the support stage deals with maintenance, logistics, and other support processes for the system of interest. On the other hand, it also deals with all processes and services that, for example, monitor the support systems themselves. This results in tasks related to e.g., maintenance or improvements. Depending on the further development of the system of interest the support systems may also have to change during its life cycle. The purpose of the support stage is to promote logistic process flows, maintenance and support measures to ensure the continuous operation of the system of interest and to provide sustainable system capabilities and services. [52]

Retirement

The retirement stage deals with the retirement of the system of interest and also includes all related systems and services that supported the system of interest during its operation. In addition, disposal is also carried out during the stage. The purpose of the retirement stage is to support the archiving and/or the dismantling of the system of interest and all support systems as well as support services. [52]

2.2.2 System Life Cycle Processes

The life cycle concept of standard ISO/IEC/IEEE 24748 as described above is enhanced in standard ISO/IEC/IEEE 15288 by relevant processes that arise in the course of a system's life. These processes are clustered into four different groups. Those four groups are (1) the agreement processes, (2) organizational project enabling processes, (3) technical management processes, and (4) technical processes [37]. An overview of the groups and contained processes is shown in Figure 12. All the shown processes are of equal importance for a holistic consideration of a system of interest along its life cycle. All the processes are documented in more detail in [37]. However, given the focus of this thesis, only the technical processes of basic engineering and partly of the detailed engineering will be described in more detail, as these are mainly relevant for the creation of architectures, as described in chapter 3.

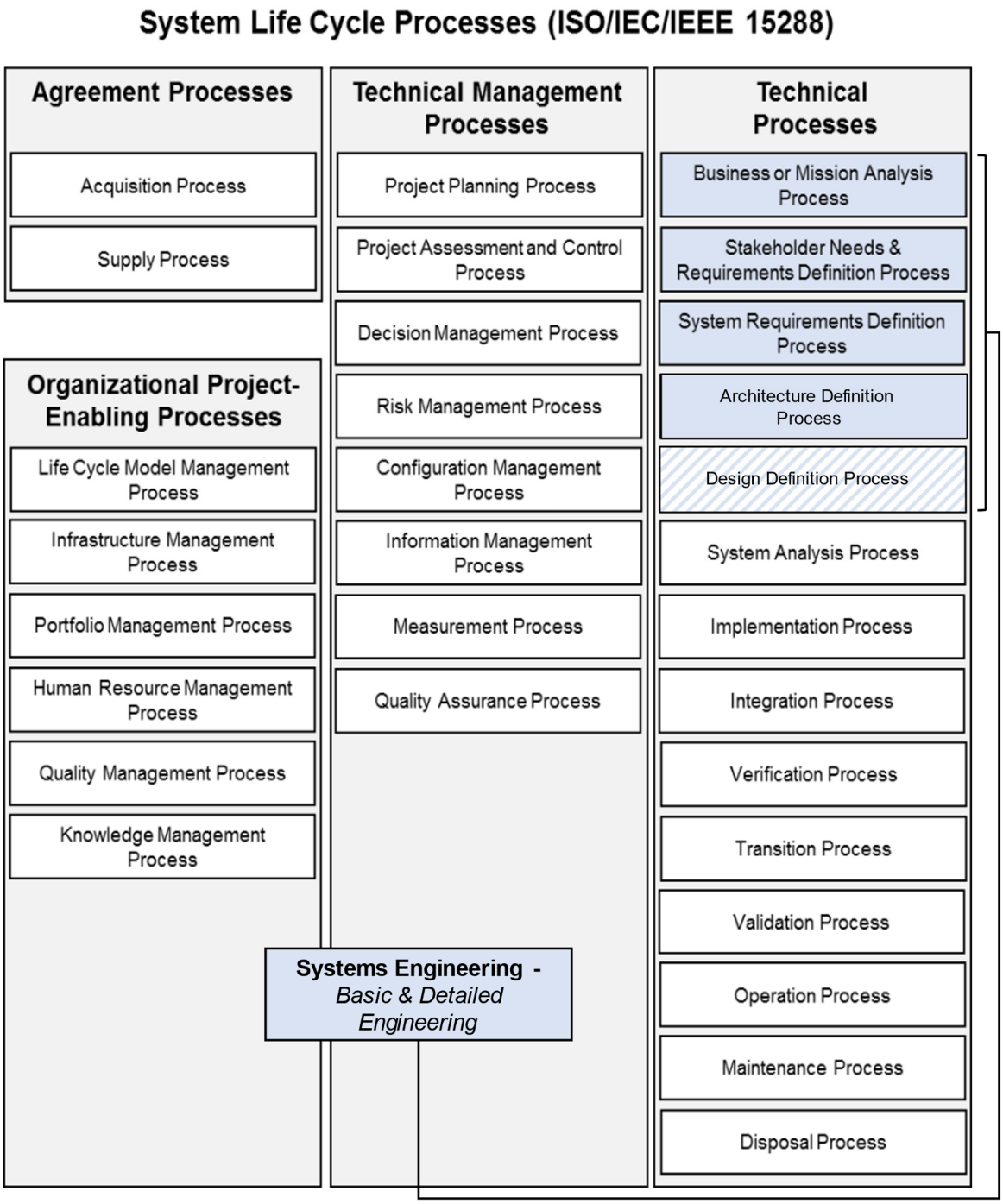


Figure 12: Typical processes within system life cycle - adapted from [37]

In the business and mission analysis process the main business or mission problem and the associated problem space are defined, the solution space is characterized, and, if possible, initial general solution ideas are derived. Building on this process, the stakeholder needs and requirements definition process defines all stakeholders and associated requirements, taking into account the context of the system of interest. Based on these requirements needed by the users within the defined environment, the system is later designed to ensure that all capabilities are available. In the next process

step, namely the system requirements definition process, the user requirements are transformed into measurable system specific requirements. The purpose of the architecture definition process is to define and determine possible architectural solutions that meet the defined requirements of the system of interest best and ultimately the needs of the users. In this process step, iterations are often performed between the previous processes and the subsequent design definition process. The results of the architecture definition process are used throughout the whole life cycle of the system. The design definition process represents the transition between basic and detailed engineering and is thus the last process considered here. Within the process the focus is on detailing system elements of the architecture to enable the later implementation of the system. [37]

The processes described above can be mapped very well to the two life cycle stages concept and development and thus delimit the main focus of this work. The described technical processes and associated stages are mainly used in the engineering of a system. For this reason, the following section takes a closer look at the engineering of a system during these life cycle stages and examines how engineering affects the life cycle of a system.

2.3 Systems Engineering

This section takes a closer look at the topic of systems engineering and introduces the connection point between the development of a system and the creation of an architecture. As a starting point, in section 2.3.1, the terms engineering and systems engineering (SE) will be defined. Based on these definitions, typical challenges in the context of (systems) engineering are described (section 2.3.2). Within sections 2.3.3 and 2.3.4, the process of systems engineering and the impact of systems engineering in the life cycle of a system as well as its effects on it are characterized. Finally, the sub-discipline model-based systems engineering and its importance as well as advantages for the definition of complex systems are discussed in section 2.3.5. An overview of the described contents is shown in Figure 13. The goal of this segment is to elaborate on the tasks of systems engineering and its effects, with the purpose of emphasizing the importance of the role of engineering within the life cycle of a system and with respect to the architecture of such a system. In addition to the following, condensed introduction to the topic of (systems) engineering, reference may be made to relevant technical literature and standards, such as INCOSE Systems Engineering Handbook [26], or ISO/IEC/IEEE 24748 [52], 3695 [56], or 42010 [30]. These sources can be used for a more in-depth introduction to the topic.

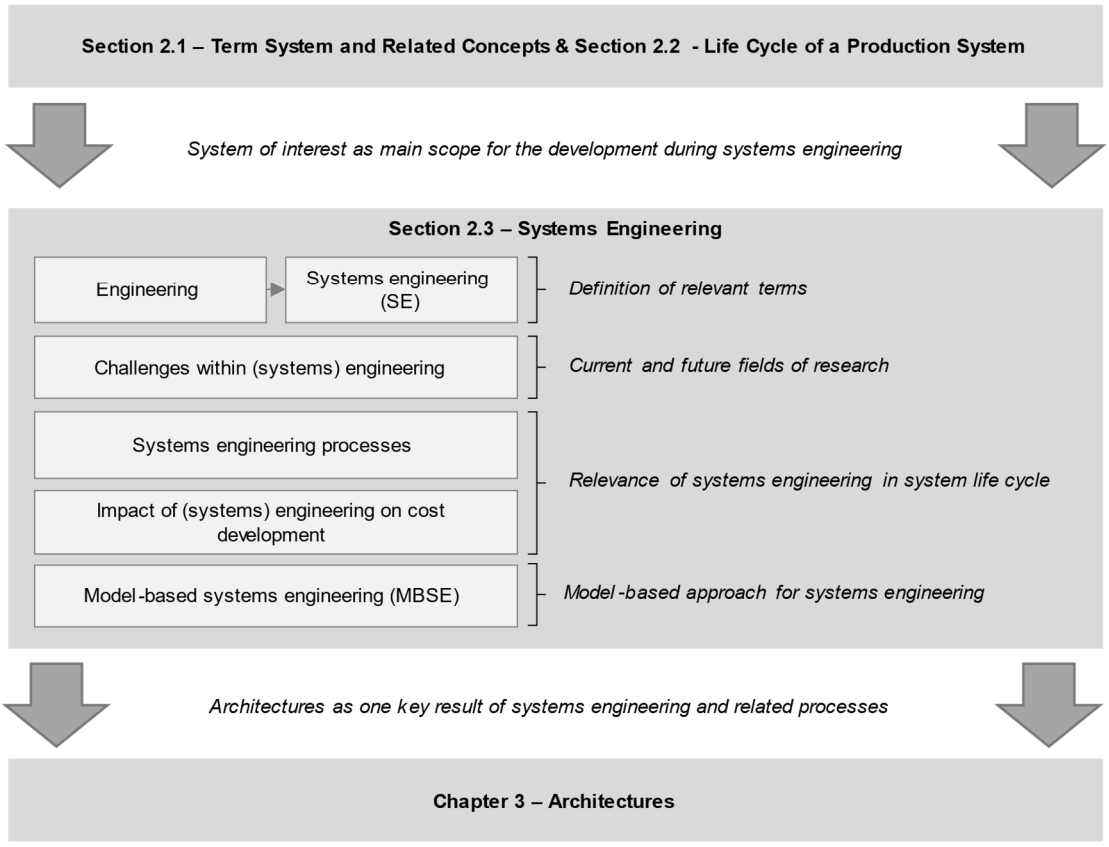


Figure 13: Overview of systems engineering state of the art

2.3.1 Systems Engineering Definition

In the following, the term systems engineering is introduced. As a starting point, and because of the interrelation to the concept of engineering, the latter will be closer examined. Based on that definition the term systems engineering is specified.

Following different literary approaches several definitions for the term engineering can be found. A detailed and comprehensive description of the term engineering is coined by the Engineers Council for Professional Development [57], specifying the term as "scientific principles to design or develop structures, machines, apparatus, or manufacturing processes, or works utilizing them singly or in combination; or to construct or operate the same with full cognizance of their design; or to forecast their behavior under specific operating conditions; all as respects an intended function, economics of operation and safety to life and property"[57]. The Institute of Electrical and Electronics Engineers (IEEE) defines engineering as “[the] application of a systematic, disciplined, quantifiable approach to structures, machines, products, systems, or processes” [41]. The IEEE definition is a simplified formulation of the Engineers Council for Professional Development definition. Although not all aspects are considered, the definition is easier to understand and sufficient for the purpose of the general explanation of the term in the context of this work. Therefore, the definition of the IEEE is considered relevant in the following.

Engineering

“The application of a systematic, disciplined, quantifiable approach to structures, machines, products, systems, or processes” [41].

The superordinate discipline of engineering can be divided into different subdisciplines based on their specific focus [21]. That means, for example, software engineering focuses on software systems and manufacturing engineering on factories [21]. Some Engineering disciplines inherit special roles, for example quality and safety engineering, as their focus can be relevant in different systems (e.g., software systems and factories) [21]. Since the focus of this work is on engineering and, in particular, on the creation of architectural descriptions of complex production systems, this chapter focuses on systems engineering and the sub-discipline of model-based systems engineering. The following chapter will be considering the discipline of architecting. Therefore, no other subdisciplines of engineering are further examined or the dependencies among themselves considered.

Following [2, 21, 22, 37, 58, 59] systems engineering (SE) can be defined as interdisciplinary and integrative approach for successfully enabling realization, use, and retirement of complex systems in their entirety within their life cycle. The task is to make the development process as effective and efficient as possible in order to achieve a high quality of results at reasonable costs [22]. In addition to the procedure itself, within systems engineering a special focus is placed on enablers such as methods, structured tools, processes, and people [2, 21]. Due to this holistic system consideration, system engineering also focuses strongly on the consideration and integration of the different disciplines, projects, and domains involved in the enabling of complex systems with the goal to improve communication and cooperation among the various parties, to ensure a successful realization, use, and retirement of a complex system [37, 58, 59]. The definition of Systems Engineering by the International Council on Systems Engineering (INCOSE) summarizes all relevant main aspects described above and is applied in the remainder of this thesis.

Systems Engineering (SE)

"Systems Engineering is a transdisciplinary and integrative approach to enable the successful realization, use, and retirement of engineered systems, using systems principles and concepts, and scientific, technological, and management methods" [58].

Systems engineering can only be carried out in an efficient and effective manner if the organization and all relevant stakeholders consider SE as useful and beneficial. In order to use SE properly within an organization, among others things, communication on SE benefits needs to be improved, SE has to be integrated into projects, needs all relevant authority and management commitment, as well as alignment of SE-related functions across supply chain and the organization [18].

For completeness it should be mentioned that the application and research in the field of systems engineering has led to the establishment of different and partly overlapping norms and standards over time [60]. These consider a variety of different aspects and are not further detailed in the context of this work. An overview of the most important norms and standards is presented in [60] and [16].

In addition should also be mentioned that in comparison to traditional systems engineering processes/procedures presented in the following sections, the topic of agile systems engineering is increasingly being considered in industry and research. Since that topic is a separate area of research with its individual advantages and challenges, it is mentioned in the context of the state of the art of engineering for the

sake of completeness, but not considered in more detail. Sources regarding agile engineering are manifold and include, for example, [15, 61–63].

2.3.2 Systems Engineering Challenges

Currently the domain of systems engineering is undergoing another significant phase of change, posing a wide variety of challenges for people and machines. Known challenges, which have already been overcome are described in [1]. Main contributors to the current shift taking place in systems engineering are market-based changes such as rising globalization [4], an increased number of competitors [5], changed customer needs for high quality but individual systems [4, 7, 8], and a continuous demographic change [4]. In addition to market changes major factors triggering change are technological developments/trends such as Industry 4.0 (I4.0), cyber-physical systems (CPS), and Internet of Things (IoT) [1], as well as global challenges in highly regarded areas such as healthcare [15]. These abstract changes are reflected in concrete problems such as adjustment of lot sizes (up to lot size 1) and the mass customization of products [10], which ultimately lead to an increasing complexity of the considered systems as well as products and place additional requirements on the design and operation of these systems [2, 3, 11–13]. Such system requirements consider, among others, flexibility as well as dynamic and intelligent system behavior [4]. Therefore, in the future, engineering is forced to focus on the implementation of intelligent/smart systems with an emphasis on adaptability, integration of relevant stakeholders in business and value creation processes, efficient utilization of resources, and (ergonomic) design of systems and processes [10, 23, 24]. All of these changes have a direct or indirect impact on the discipline of engineering and represent the intersection between the current state of the art of engineering and the current and future fields of research. In order to continuously improve and develop engineering, these changes and resulting challenges must be addressed and solved in the future. Some of the most common challenges addressed in literature and in daily business are listed in non-particular order below.

- Complexity: One of the main challenges is to align and evolve systems engineering with the increasing complexity of systems and products as well as related production and work processes [1, 3–5, 22]. This increase in complexity is caused by different factors such as regulations, technology developments, compounder requirements, increased functionality, the stronger linking of system elements, and their increased interactions [4, 13, 16, 22]. Due to the increased complexity of the system and the associated development processes, the goals set in development projects, such as time, costs and quality, are

increasingly not achieved to the planned extent [22]. In addition, it should be mentioned, that complexity can be of different types, which will have different influences. In source [64] a comprehensive overview of different types of complexity is provided.

- Uncertainty: The increasing complexity as well as occurring constant changes in the environment of a system in conjunction with innovations with respect to processes and materials lead to uncertainty [14, 15]. Those circumstances make it difficult for the involved stakeholders to predict and design required functionalities and capabilities of a system of interest [14, 18]. The successful management and control of uncertainty is important for the stakeholders to develop predictable and dependable systems [65]. Uncertainty that is not taken into account or occurring spontaneously can have a negative effect on the system and its performance possibly leading to necessary changes as well as additional expenses [15].
- Life Cycle Management: Further challenges arise from the situation that, on the one hand, due to more requests for specialized products/systems the life cycles are becoming shorter and more dynamic and, on the other hand, different systems with different life cycles are used in combination in engineering [1, 16]. This places demands on both the development process of a system with respect to cost, quality, and time as well as on the management and control of the different life cycles [1, 16]. In addition, the large number of variants that arise must be managed and administered [16].
- Project Management: In the context of changing system development, the project management itself also presents certain challenges. Important points, which intensify due to the described changes, are a good and simple entrance for all stakeholders into the project [17], which is a basic prerequisite for recognizing and integrating all disciplines and enabling the necessary interaction between relevant stakeholders [17]. Another important point, despite factors such as complexity and uncertainty, is to correctly estimate the effort in terms of costs and time in order to be able to successfully carry out and implement the system development project [18].
- Integration: The described increase in complexity, uncertainty, advancing globalization, and the possibly distributed engineering efforts might lead to insufficient cross integration [18] from which challenges arise, such as in the area of the involved stakeholders and disciplines [5], the specific artifacts, and the tools used [19]. Many of the previously separate and highly specialized disciplines and stakeholders within this environment use specific approaches

and procedures [16]. In order to overcome the current status and to achieve successful integration, a fundamental understanding, management of interfaces, as well as a cross-project, cross-discipline and cross-company specific approach must be created and supported by appropriate norms and standards, for example, with respect to communication and collaboration [3–5, 16, 20, 21]. Depending on the task, tools and processes used, it does not always make sense to work simultaneously [17]. Therefore, in addition to the factors already mentioned, it is important to achieve a good compromise between interaction and isolation of the individual stakeholders, disciplines, companies or subprojects in order to achieve a defined goal as efficiently and effectively as possible [17]. Other challenges that can be located in a similar environment concern the consistency of roles and tools [17] and interoperability, for example between systems [15].

- Knowledge Management and Reuse: The issues of knowledge management and the reuse of existing solutions, which result from the dynamics and short lifetime, also pose certain challenges in engineering [17, 20]. The challenge in engineering is not only to create viable solutions, but also to describe them in a meaningful way, to store them and, if necessary, to pass them on and present them in a form that requires less additional effort [17].
- Workforce: Another area of consideration, which is not solely limited to engineering but is relevant and influenced by all the changes described above, is concerned with the workforce. Due to the trends described, such as increasing complexity or the current global Covid-19 pandemic and its consequences, the complexity of the work tasks is also changing, as are, in part, the requirements for the qualifications of the affected stakeholders and their workplaces [4]. Temporal and location specific flexibility, as well as the individual analysis of skills and the targeted use of the affected stakeholders is becoming increasingly important [4]. In addition, there are also challenges resulting from demographic change and the increasing average age of the stakeholders [4].

Most of these challenges, if not addressed, lead to reputational loss and take away from planned profit margins [18]. Systems engineering is not only beneficial on a technical level and within a company, but also on a business level [18]. In addition to the general challenges for systems engineering that are relevant in many areas and described above, other individual challenges arise depending on the focus of engineering. For example, for the engineering of future production systems, among

others, challenges would arise from topics such as Industry 4.0 or cyber-physical systems. These specific challenges are not considered in detail in this thesis, since they are expressed individually in relation to the specific application scenario. However, further details on the challenges and fields of action in I4.0 and CPS/CPPS can be found, among others, in [4, 9, 12, 66–69]. In addition to the challenges currently being researched and implemented, possible future engineering challenges are already described and discussed, i.a., on the basis of occurring trends. The renowned International Council on Systems Engineering – INCOSE presents, for example, its vision of systems engineering for 2025 and describes how engineering will change in the future and which challenges have to be faced [3]. Generally speaking, [3] states that systems of the future

- “need to respond to an ever growing and diverse spectrum of societal needs in order to create value”.
- “need to become smarter, self-organized, sustainable, resource- efficient, robust and safe in order to meet stakeholder demands”.
- “need to be engineered by an evolving, diverse workforce which, with increasingly capable tools, can innovate and respond to competitive pressures”.
- “need to harness the ever growing body of technology innovations while protecting against unintended consequences”.
- “need to be aligned with global trends in industry, economy and society, which will, in turn, influence system needs and expectations”.

To improve managing current and future challenges, approaches such as systems engineering and related disciplines like model-based systems engineering are needed [22], which focus on interdisciplinary and system comprehensive methods for the development of systems [16]. If this necessary shift is not processed seriously, the differences between established discipline-focused development methods and the required development methods will worsen due to the increasing complexity and make it much more difficult to engineer a system in the future [16].

2.3.3 Systems Engineering Process

This section takes a closer look on the process of engineering a system. Over time, from experience and change, different processes have emerged on how a system of interest can be developed within systems engineering. An engineering process basically describes the application of the engineering discipline and can be defined as follows.

Engineering Process

An engineering process is defined “[…] as a sequence of activities of creative application of scientific principles to design or develop structures, machines, apparatus, or manufacturing processes; all as respects an intended function, economic and safe operation” [70] following [71].

The results of systems engineering and the associated processes are called engineering artifacts [61, 72, 73]. These are referred to simplistically as artifacts in this thesis. The definitions differ slightly depending on the engineering domain but can be expressed and applied generally. ISO/IEC 19501 defines an artifact as a representation of "[...] a physical piece of information that is used or produced by a software development process” [74]. VDI 3695 defines the term artifact, despite the specific focus on industrial plants, more generally and as follows.

Artifact

An artifact refers to all "tangible and intangible project deliverables" [56]. Tangible artifacts refer to, for example, "engines, pumps, [or] functional modules" [56]. Intangible artifacts refer to, for example, "plans, architectures, [or] specifications" [56].

It should also be noted that artifacts are not only created during an engineering project or during the engineering-related life cycle phases of a system. Artifacts are also created during all other actions related to the system of interest and the remaining life cycle phases of the system. These artifacts can have relationships among themselves through their creation process and inclusion of information from other projects/life cycle phases. Typical created artifacts of the engineering process, like requirements or architecture, are described in more detail by, among others, [5] and [75]. Depending on the variety of engineering processes, artifacts differing in name and content may be created. Simplified, the space of possible relevant input which is considered during the development of a system of interest and related artifacts can be considered as design space [76]. In general, it can be stated that, amidst others, due to the increase in complexity, systems engineering and associated engineering processes run through a dynamic and iterative sequence of process steps in a relevant design space to define a system of interest and create related artifacts [76]. [77] describes that iteration is a core aspect of product/system development [78, 79], both at the macro and micro

levels [80]. According to [76], a design space can be defined by the three dimensions (1) "from the abstract to the concrete, [(2)] from the general to the detail, as well as [(3)] through views or aspects [...]" [[76] and [81] in [76]]. This possible classification of the design space is presented in Figure 14.

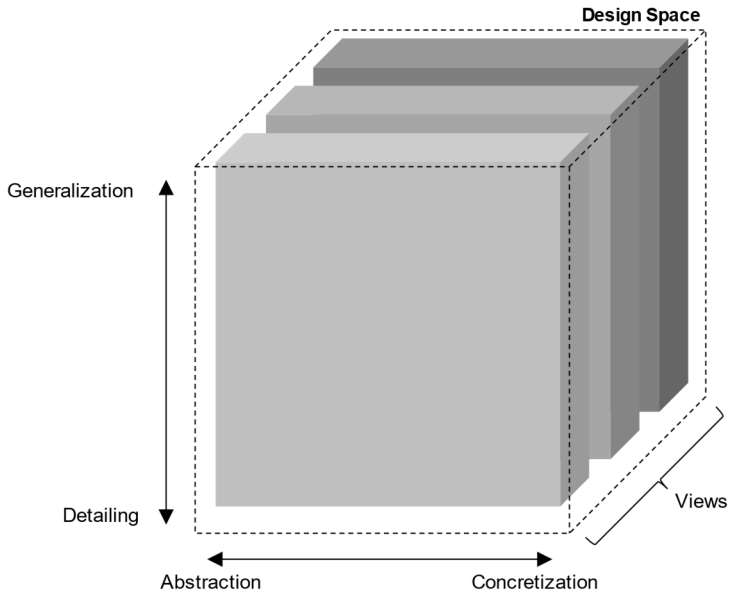


Figure 14: Dimensions of design space adapted from [76]

Due to the dimensions of the design space, the typical engineering process results in a combination/interaction of analysis, synthesis, and consideration of different views [76, 77]. The views help to emphasis specific aspects of the considered system [77] (see section 3.1.1.2 for definition of view). The possibility of considering different levels of detail during the engineering process enables the use of both top-down and bottom-up approaches (for a definition, see section 5.4.3.1) [76]. Despite the multitude of factors and the resulting possibilities, [76] defines that "[...] there is a basic approach for all types of development tasks: from the abstract to the concrete, from the general to the detailed and from the functional to the structural". In addition, it should be stated that all engineering processes are based on different prerequisites regarding methodologies (e.g., goal definition, identification of boundary conditions) as well as procedures (e.g., abstraction and decomposition), which have been summarized extensively by [77] and are therefore not considered in detail within this thesis. The actual, case-specific, engineering process results from the concrete conditions, application scenario, and depending on the system of interest [76] and is following a certain engineering strategy, like a waterfall or spiral approach [82]. Due to that nature a wide variety of both standardized and in research and industry recognized engineering processes as well as individually developed procedures emerged.

According to [18, 70, 83], the former includes, among others, ANSI/EIA 632 – Processes for Engineering a System [84], ISO/IEC/IEEE 15288:2015- Systems and software engineering - System life cycle processes [37], VDI 2206 - Design methodology for mechatronic systems [25], VDI 3695 - Engineering of industrial plants - Evaluation and optimization [56], which can be assigned to different businesses, such as product, component, and solution business. Due to the large number of processes, there are potentially several relevant engineering processes that could be applied in a given application scenario based on their consideration of similar contents. However, based on the specific characteristics of the processes or their status as an established process in a particular domain, the most suitable process for the considered application scenario should be selected. As stated above, literature provides several sources in which engineering process of certain domains or types are documented and put in relation with each other. Based on the comprehensive representation of various engineering process by [70], in this thesis, the procedure described in VDI 2206 was selected as a representative example of an engineering process, which is explained in more detail in the following section.

2.3.3.1 VDI 2206 as an Example of Engineering Processes

In order to detail the individual steps of an engineering process and relate the thesis topic of creating an architecture description to the concepts of systems engineering as well as the project and system life cycle phases, the standard VDI 2206 is described as a representative example. The VDI 2206 describes the development of mechatronic systems, such as production systems [25, 70]. According to [25], three key elements represent the development process of mechatronic systems; (1) a problem-solving cycle at the micro level, (2) the V-model at the macro level and (3) process modules for recurring development steps. A problem-solving cycle (1), as presented for example by [85] in [25], regulates and organizes the procedure in the engineering process [25]. The cycle is designed in such a way that it can be applied continuously and enables flexible processing of subtasks and solving of occurring problems in the engineering process [25]. The V-model (2), also called Systems Engineering VEE, describes the main development steps, their interrelationships, and their sequence in the engineering process [86, 87] in [25]. The process steps defined in the V-model are partially processed by the problem solution cycle, as already described [25]. For the processing of recurring tasks, predefined process modules (3) are used [25]. For the representation of the relationships between life cycle and project phases as well as the relevant steps of engineering in relation to the creation of an architecture, mainly the macro cycle and thus the so-called V-model is relevant and will be described in

more detail. For a visual representation of the V-model concept see Figure 15. For more details on the problem-solving cycle as well as predefined process modules see [25].

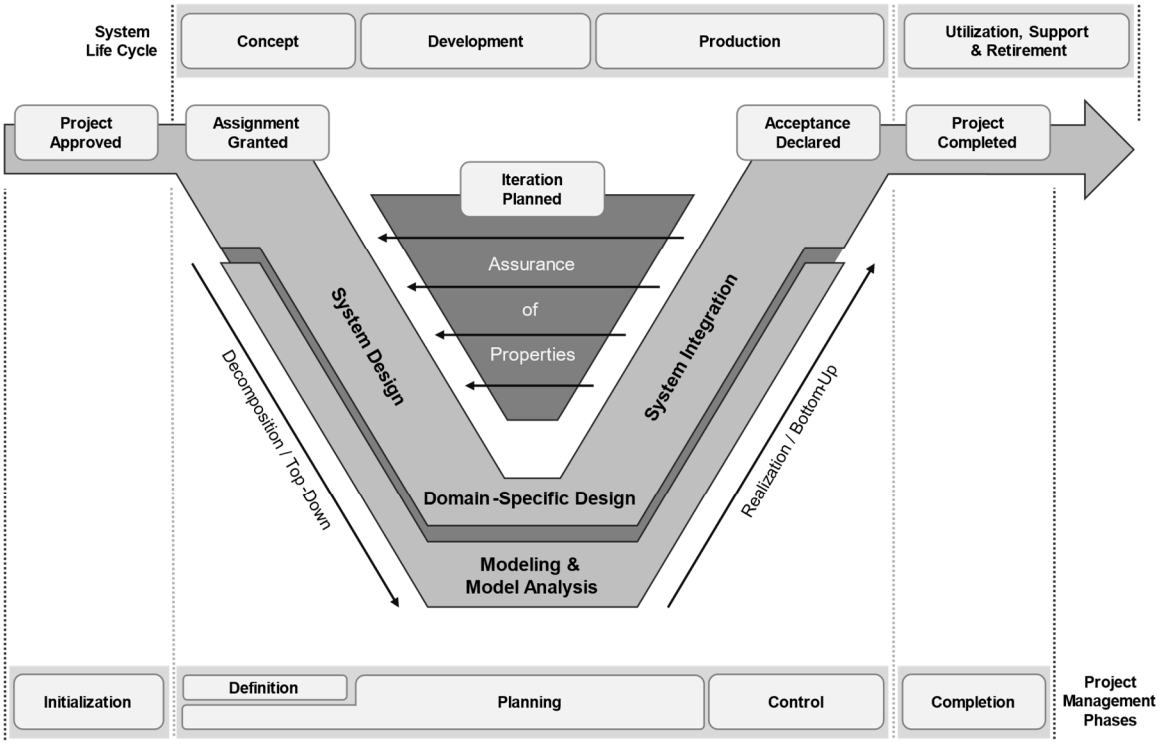


Figure 15: V-model as exemplary systems engineering process representation - adapted from [25, 59, 75, 88–91]

The procedure for the design of a system represented by the V-model can be divided into different steps. Upfront, it should be mentioned that during the development of the system utilizing the V-model different life cycle and development project phases are passed through in parallel, which are also described and related to each other in the following. Before starting with the top-down decomposition [89, 91, 92], at the beginning of each development, a specified need and problem, interfaces and involved stakeholders, as well as a clear concept for system development have to be defined [25, 28, 59, 91]. This clarification happens in the initialization phase of the project which takes place outside the V-model and ends with a key decision point within the V-model, the approvement of the project [75]. After the initialization phase has been completed, in parallel, the definition and planning phases of the project are processed [75]. At the same time, the needs and concepts of the relevant stakeholders are defined in the V-model in the form of requirements [25]. As the definition of the system begins and first design considerations are made, the life cycle of the system of interest is initialized within the concept phase [90] and the architecture of the system

commences as well [61]. The definition phase of the development project ends as soon as the assignment for realization of the system is granted [75, 88]. After complete consideration of the system and specification of all relevant requirements, which shall be used for development but also to validate the results produced in the engineering process of the system [25], the system transitions from the concept to the development phase of the life cycle. Based on this initial starting point, the actual solution design of the system is developed [25]. For this purpose, by means of decomposition and detailing, functional elements are assigned to relevant logical and physical properties which the system must fulfill [25]. During the V-model system design and the development phase of the system life cycle, the architecture of the system is fully defined at different levels of detail [28, 75, 90, 91]. Following top-down decomposition first, the overall system architecture and then, step by step, the associated details in the form of architectural descriptions of the subsystems and relevant components are defined [89]. After the architecture is created the domain-specific design for the system is created [25, 90]. In this phase the detailing and the implementation of the system begins (e.g., software and hardware specification, architecture) [28, 59, 75]. The domain-specific design is usually created separately and in relation to the domain involved (mechanical engineering, electrical engineering, information technology) [25]. During the definition process new and existing solutions should be considered and all implemented decisions should be documented for better traceability [59]. In the context of system integration of the V-model, the defined artifacts are brought together in a bottom-up approach to realize and implement the overall system solution [25, 89, 91, 92]. This is mainly achieved in the production phase of the system life cycle, as well as in the planning and control phase of the engineering project [75, 90]. In the control phase, the client reviews the progress of the project and the contractor provides the necessary development [75]. During the integration phase of the V-model, the artifacts are verified and validated by comparing the requirements and the actually available and developed solution to ensure that the results produced match the defined stakeholder needs and desired characteristics [25, 28, 59, 89, 90]. The result of a cycle in the V-model is called product [25]. In relation to this thesis it should be annotated that the term product in the V-model describes the actual developed system. When using product in the thesis and especially in relation to a production system, a product represents the result of the production process which is carried out by the system of interest. Depending on the development task at hand and its complexity, the macro cycle must usually be run through several times in order to be able to realize the system of interest [25]. This is not shown in the simplified illustration of Figure 15 but explained in more detail in [25]. It should also be noted

that with increasing complexity and by considering integration on different layers of granularity in a timely manner, a strict separation of top-down and bottom-up design becomes increasingly difficult [25]. This might lead to an iterative approach and a continuous alternation between bottom-up and top-down procedure may be necessary [[25, 93] and [94] in [25]]. Iterations are performed until the system is fully realized. During system integration, the system evolves from the development phase of the life cycle to the production phase of the cycle [90]. The control phase of the project ends and the transition to the final phase, the completion phase, begins with the declaration of acceptance of the system [75]. After the production and acceptance of the system, the system enters the utilization phase of the life cycle and the development project ends [75, 90]. A detailed representation of the project phases can be found in standard DIN 69901 [88]. The life cycle concept of a system has already been presented in section 2.2. As shown in Figure 15, the individual steps of the V-model are supported by modeling and model analysis, for example, to validate properties of the system with the aid of suitable models and tools. Model-based systems engineering is described in section 2.3.5. The modeled architecture content can also be utilized in the later system life cycle phases, for example, for supporting trainings, modifications and updates, as well as failure analysis [28]. An overview of the artifacts typically created when using the V-model can be found in [75]. According to VDI 2206, for the practical application of the model, it should also be taken into account that, for example, due to the control of the development risk, some subsystems must be developed before the overall system is developed as envisaged in the model [25]. It should also be noted that due to the constantly changing environment and the increasing system complexity, the associated processes, such as VDI 2206, must also be continuously checked for their applicability and expanded as well as adapted if necessary [95]. The shown engineering process and its decencies to life cycle and project phases can be transferred like this or in a similar way to the other engineering processes mentioned above (taking their individual features into account). A more detailed consideration of the application of different engineering processes is not shown, as that is less relevant for the focus of the thesis.

Nevertheless, it should be mentioned, if of interest, that, among others, [70, 92, 96, 97] provide a comprehensive overview/comparison of different engineering processes and point out their commonalities and relations.

2.3.4 Influence of Systems Engineering on Life Cycle Cost Development

In the following section, the cost development during the different life cycle phases of a system is considered and how the individual phases affect the development of the

total costs. In particular the impact of systems engineering from an economic point of view and especially of the early phases, when the architecture specification is carried out, will be examined. In general, it can be stated that a system causes costs. The individual phases in the life cycle of a system contribute for different percentages of the life cycle costs of the system. The life cycle costs, from 0 percent at the beginning to 100 percent at the end of the cycle, are shown cumulatively in Figure 16. All the given numbers within the figure might change from system to system, but are accepted and mostly, with minor variations, represented similarly in literature. The actual increase between the individual life cycle stages is represented by the difference between the two compared stages.

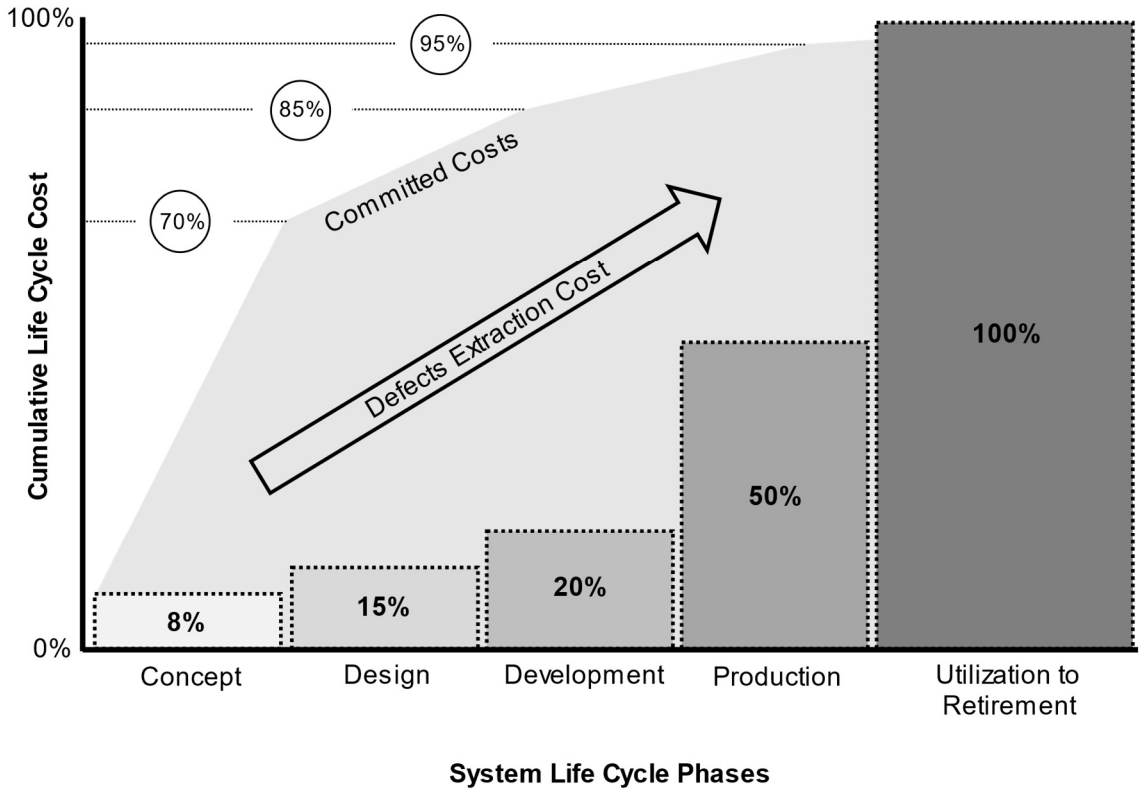


Figure 16: Committed and actual generated cost in system life cycle - adapted from [26]

The life cycle phases in the figure are not shown to scale. Rather, the phases in which there are large discrepancies between actual and committed costs are shown in more detail. These are mostly the early phases of the system life cycle, like concept and design, in which the creation of architectures, the main topic of this thesis, has a major impact. Later phases (e.g., production and utilization) are summarized in thematically matching columns, as they do not have such a great relevance on the overall cost development with respect to determination of committed cost. In contrary to the previously introduced life cycle phases, the additional phase of design, previously not

represented, is shown on purpose. This allows to map the influence on the development of costs in even greater detail. The design phase can be, with respect to the introduced life cycle concept of section 2.2, thematically added to the development phase. With respect to the introduced macro-process of systems engineering (see section 2.3.3), the concept phase considers the stakeholder needs and system requirements, the design phase the basic engineering and the creation of the architecture of the system, the development phase the domain-specific detailing of the architecture description, and, as shown within the V-model, the production phase the bottom-up integration.

It can be clearly seen, with respect to Figure 16, that there is a strong discrepancy between the actual cost and the costs already committed at the same point in time. Committed costs describe costs that are determined by decisions during one or more life cycle phases but will become cost effective at a later point in time, at which the decisions made are realized and the required efforts are converted into actual life cycle costs. The discrepancies are particularly significant in the development phases/the early phases of the system life cycle [26, 98]. Since fundamental decisions for the later life cycle phases are made during the early phases, for example, specification of architecture, only low actual costs are incurred, but a very sharp increase in committed costs happens. After the concept phase in which approximately 8% of the actual costs are incurred, one is committed to almost 70% of the absolute system life cycle costs. During the design and early development phase, the committed costs increase further to around 80 percent, while at the same time actual cost of approximately 15 percent are created. After that, the actual costs increase steadily and the committed costs increase only slightly. Because a lot of the effort within the early development phases is of a conceptual nature and the actual physical implementation and use of the predefined concepts takes place in the later life cycle phases, costs can be influenced very well in the early phases and very poorly in the late ones [99]. This decrease in influence and the increasing rigidity of systems as they move through the life cycle phases means that the costs of implementing changes or resolving errors increases in the opposite direction from concept phase to retirement phase [[26], [100] based on [101]]. The cost multiplier in relation to the total cost is still relatively low in the early life cycle phases, but then rises very sharply in the direction of system utilization. Errors that are only detected at a very late stage thus lead to a sharp increase in total system costs. This factorization of costs is shown in Table 1. Overall, engineering and the considered topic of architecting account for system cost factors from concept to production phase (multipliers of 1 to 78), when considering the V-model and its phases as a basis.

Table 1: Cost factors/problem solving adapted from [18]

Life Cycle Phase	Systems Cost Factors
Concept	1x
Design	3-8x
Development	7-16x
Production	21-78x
Utilization	29-1615x

As a result, the impact on cost of engineering and architecting is relatively high in the early phases of the system life cycle but at the same time the cost for elimination errors or conduction changes relatively low compared to the other life cycle phases.

These two factors, the high committed costs in the early phases and the multiplication of the costs for error maintenance/changes in the later life cycle phases, show the importance of efficient and effective engineering and architecting in the early life cycle phases. Solving problems early in the life cycle of a system only slightly increases costs [18]. Troubleshooting later in the life cycle increases costs many times over [18]. Due to this dynamic the aim is to make required changes at early life cycle stages rather than at later ones to reduce the total life cycle costs and the cost multiplication impact. The shift of adaptions to the early stages will require additional development efforts, for example in the form of architecture frameworks or blueprints like reference architectures, and consequently will result in initially higher costs. The impact on costs that can be exerted by the application of more effective and efficient system engineering and architecting, compared to current widespread practice, is shown in Figure 17.

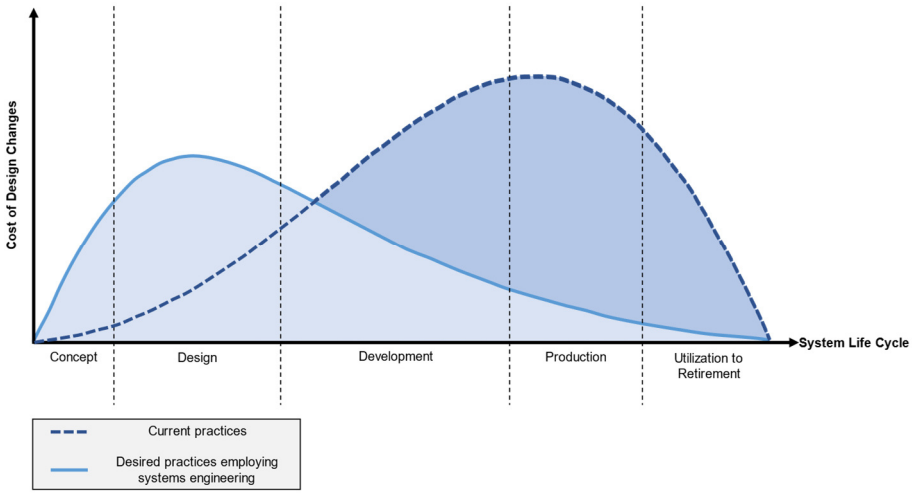


Figure 17: Cost of design changes with respect to system life cycle phases- adapted from [18, 26, 102]

2.3.5 Model-based Systems Engineering

As already described in the introduction, different circumstances lead to challenges in the context of system development projects, carried out by different disciplines and across organizations [1, 15, 92]. This has led to an ongoing trend of supporting development projects with model-based approaches [1, 92], “[...] for improving the efficiency and effectiveness of systems engineering through the pervasive use of integrated descriptive representations of the system to capture knowledge about the system for the benefit of all stakeholders” [103]. The evaluation of that trend has been the topic of several surveys, like [104], which are considering the current and the future status of MBSE. By "[using] models, the system or product to be developed can be better understood, analyzed, detailed, documented, communicated, and transmitted for further processing" [92]. In order to do so, it is required that data can be exchanged effortlessly during the system design process [15]. It also shows that good collaboration is essential for the successful realization of a system [15]. In the traditional approach of systems engineering, mainly textual and design-related documentation is created during the development of a system, which is why the approach is also referred to as a "document-based" approach [105]. Such a document-centric engineering approach requires human-readable documents and relies heavily on the knowledge, creativity, and ability of the stakeholders involved to mentally integrate the loosely connected document content [15]. In contrast, model-based systems engineering (MBSE) represents a "model-centric" approach and focuses mainly on the creation of a coherent system model for the system development, which consists of all system relevant information, such as requirements, design, and verification information [105, 106]. The design of architectural content is structured around these models and their content is step by step refined along the life cycle of a system of interest [103]. MBSE in comparison to the traditional SE does not challenge or replace the established systems engineering principles, but rather enhances their impact, for example, through a strong focus on consistency due to the shift from document to model centric principles [15, 107]. The generated model reflects the main outcome of the MBSE and, in contrast to traditional systems engineering, the creation of documentation takes a secondary role, since this information can ideally be generated from the model [105, 107]. The advantage of the approach in comparison to a document-based one is that the model content slowly replaces the individual documents, that evolve individually over time and creating inconsistencies which should be avoided at all costs [103]. MBSE represents a long-term model-focused development that is being embraced by other relevant engineering disciplines, for example, mechanical engineering [108]. It provides a good foundation for managing the process of developing systems as it becomes accepted and is applied in the

industry on larger scale [22]. The goal of using an integrated cross-discipline system model throughout MBSE is to reduce complexity-related misunderstandings, highlight the effects of changes more clearly, and enable tracking of changes across discipline-specific models [21, 109]. This does not mean that those goals are not relevant within traditional SE, but rather that MBSE makes an important contribution to meeting these challenges [21]. For example, traditional systems engineering follows the task of integrating different disciplines, but MBSE particularly focuses on strengthening the cooperation and communication between the disciplines [21, 22] and follows a more comprehensive methodological approach in terms of capturing, integrating and maintaining engineering activity results [105]. The earlier MBSE is used in the SE life cycle, the greater the positive effect and the benefits that can be achieved [15]. In summary, Model-based systems engineering (MBSE) can be defined as follows:

Model-based Systems Engineering

"Model-based systems engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation, beginning in the conceptual design phase and continuing throughout development and later life cycle phases" [39, 107].

The main idea behind MBSE is that a model is defined at the beginning of the system development life cycle, which then grows through all phases and evolves from early decision making to the complete system architecture [15]. Therefore, MBSE focuses specifically on the formalized modeling of a system of interest to support all activities along the life cycle of the system [110] in [105]. A model can be characterized as follows.

Model

"A model captures a view of a [...] system. Hence, it is an abstraction of the physical system with a certain purpose; for example, to describe behavioral aspects of the physical system to a certain category of stakeholders. A model contains all the model elements needed to represent a physical system completely according to the purpose of this particular model. The model elements in a model are organized into a package/subsystem hierarchy, where the top-most package/subsystem represents the boundary of the physical system" [74].

A model represents a simplified version of a coherent and consistent set of elements, components, or a complex system [[25, 38],[49] in [39]]. In the model the current, past or future reality is represented in an abstracted form and with an actual or ideal status of the system [111]. A model is defined as "[an] abstraction of a system of interest constructed from one or more representations. The [overall system] model is the total recorded knowledge of a project [...]" [39]. A representation defines "[a] partial description of the system of interest [...]"[39]. Within this thesis a representation will be referred to as a view. A view may represent a text, table or diagram described in, for example, a graphical, mathematical, or natural language [39]. Different views represent a specific viewpoint [39]. A comprehensive definition of viewpoint and view can be found in section 3.1.1.2. The system model can be considered one of the central artifacts of MBSE [110] in [105]. The term artifact has already been defined in section 2.3.3. As part of the model-based approach, the contents of the model are then either supplemented, adapted or used by all those in a relevant relationship to the system [106]. The term model element is defined as follows.

(Model) Element

"An element is an atomic constituent of a model" [74] and "[a] model element is an element that is an abstraction drawn from the system being modeled" [74].

Besides the elementary step from document-driven to model-focused system development, MBSE also requires an adequate process that extends and complements the traditional systems engineering processes and integrates the use of the model [39]. In this context some processes have already been created and standardized among others, INCOSE - Object-Oriented Systems Engineering Methodology (OOSEM) or Weilkiens - Systems Modelling Process (SYSMOD)) [39].

2.3.5.1 Benefits and Implementation Challenges of MBSE

Systems Engineering, as described in section 2.3.1, generates independent documents due to the document-centric approach, the contents of which may overlap and are thus loosely connected with each other [39]. Due to these characteristics of the SE results, it is difficult to perform complete evaluations of the content against, for example, defined quality criteria [39]. The use of a model in MBSE as central connection point, aims to enable easier assessment and verification of complex relationships between content and address such challenges through the use of consistently related elements within the model [39]. In simple terms, model-based systems engineering enables

better data handling, such as the collection, evaluation, and maintenance of system data [107]. Therefore, the application of MBSE within the process of engineering systems results in different advantages, which are listed below:

- Enhanced and clear communication between relevant stakeholders and across disciplines within the life cycle of a system of interest [[28, 39, 107, 112] & [110] in [105]].
- Mastery of system complexity and improved management of development risk due to an integrated system model and the ability to view specific aspects of the system from different perspectives, for example in the context of changes [[92, 107] & [110] in [105]]. This impacts, for example, the improvement of estimation of costs and the reduction of errors related to testing and integration [39].
- Optimized quality due to the opportunity to check, for example, correctness and completeness on the basis of the system model [[28, 107, 112] & [110] in [105]]. This enables quality enhancements, for example, through improved specification, allocation and tracking of requirements, assessment and selection of architectural options, or the creation of consistent and sustainable documentation [26, 39, 112].
- Assurance of a standardized knowledge transfer, the reuse of information and as a consequence the possible reduction of efforts for the adaptation of model contents [92, 107].
- Increased productivity is enabled, for example, through joint content definition and reuse, optimized team collaboration, and automated document creation [39].
- Activities related to the system can be automated better, since the data used is machine-readable and can be processed more easily in relevant tools [106].
- The realization of continuous tool chains is enabled by the possibility to generate new input data and by the improved data transfer, which does not require manual re-entry into tools along the tool chain [106]. Generated data can take different forms, such as code or documents [106].
- Due to the integration into models and continuous tool chains data is up-to-date in most cases and can be used across different disciplines at the same time [106]. In certain constellations, inconsistencies may occur, which can be prevented by taking appropriate precautions within the tools.
- Beneficial starting point for the transfer and teaching of SE fundamentals through representation of relevant approaches within models [107].

Further purposes and advantages of MBSE are, among others, mentioned by [26, 92].

In addition to solving technical challenges, the consideration of non-technical challenges is crucial for the successful implementation of a MBSE approach [103]. Before the benefits of MBSE can be achieved, a culture change must take place, which is one of the main starting points required for a successful transition from the document-centric to the model-based approach [113, 114]. Similar to the application of systems engineering, MBSE requires some enablers concerned with topics like appropriate processes, required tools, utilization of standard techniques and representations, risk tolerance, easy transition of knowledge, and training to relevant people, in order to achieve a successful implementation within the organization [39, 103, 113]. In most cases the corporate structures of an organization do not meet the prerequisites for MBSE to be implemented without adjustments to these structures [[110] in [112]]. To this end, all affected stakeholders and in particular the management must be prepared to accept and support the goals of MBSE deployment, the corresponding additional costs, changes to their task profile, and infrastructural changes [15, 103, 115]. Previously failed implementation attempts can represent additional barriers to entry [15]. Concrete challenges that must be mastered are, for example, the change from the previous document-based way of working and thinking to a model-based way of thinking and the associated necessary differentiation between the presentation of information and the actual results [15]. Another challenge is the learning and application of modeling tools and languages within MBSE, which can be a barrier to entry for stakeholders [15]. If this change has already taken place, there are still some challenges, such as digitalization, integration of models, as well as maintenance of tools and modeling languages, that need to be addressed during the use of MBSE [15, 115, 116]. As this work is not concerned with optimization and preparation of organizations for the effective implementation of concepts like SE and MBSE it is assumed, that, like SE, MBSE is accepted all over the organization by all relevant stakeholders and considered a value-adding concept.

2.3.5.2 Utilization of MBSE along the SE Life Cycle

The long-term vision for the MBSE approach and its system model is to use them as a basis for all relevant activities along the system life cycle. At the beginning of the development of a system a system model can be used primarily to describe the basic initial situation, such as capabilities, goals, and context. Specifically with the help of modeling, the system context, and its relevant components; interfaces, requirements, and highly simplified architectures are developed in the form of models. In the next phases and after the development of the system architecture has been completed, the overall system model is gradually further detailed and integrated into the realization

process of the system. Particular attention is paid to the possibility of testing (partial) models, as well as distributed processing and integration, since these capabilities are indispensable in the context of cross-disciplinary processing. With the ability to test on the model, potential impacts can be evaluated better and consequently efficiency is increased. After the system has passed the planning and realization phases of the life cycle, models can be used during operation and maintenance to provide the affected stakeholders with specific information about the system and its properties or interfaces. Information from the models can also be used for troubleshooting or warranty cases as well as for the further development of future products. After the operating phase of the system, it enters the decommissioning phase, in which the models can serve as a basis of information about, e.g., potentially hazardous materials. In order to operate a sustainable knowledge management and to keep documentation, the models have to be archived accordingly at the end of the system life cycle. [15]

The methodologies and applications of the model will continue to evolve as the acceptance and use of MBSE increases [15]. Further information on the use and developments of MBSE along the system life cycle can be obtained from the various interest groups that are involved with the evolution of the topic. One of these interest groups is for example the INCOSE Model-based Conceptual Design Working Group [117].

2.4 Summary – Engineering of Production Systems

To summarize, the key points of engineering of production systems are that a system of interest has a structure of system elements/subsystems that represent the granularity of the overall system. Each system is to be classified in an associated environment with the related system context introducing the interfaces to other interacting elements. During its life, the described system passes through different phases from concept through utilization to retirement. In these phases, different requirements are placed on the system itself as well as on the associated processes and stakeholders. In the context of this work the focus is on the early life cycle phases during the engineering of a system (concept, development phase) and in particular on the architectural description of production systems. These systems are used to create and provide products of a specific form, fit, and function. This specification affects the selection of possible engineering processes for the creation of such a system. In simple terms, engineering deals with approaches and methods of how such a system can be developed, designed, implemented, and, if necessary, adapted to meet the current challenges in this area as efficiently and effectively as possible. During the engineering

process, different results/artifacts, for example, a system architecture description, are created under consideration of the specific design space. In general, it can be summarized that during an engineering process several steps are passed through. First the initial situation is considered and relevant requirements are defined, then the basic engineering with the design of the architecture is carried out where the domain-specific detailed design as well as the system implementation with subsequent commissioning are considered. During the process, the partial results created should be compared continuously with the defined requirements to support the creation of the system and to avoid inconsistencies and risks during implementation and utilization. This approach is increasingly being extended with model-based approaches, which, among other things, can have complexity-managing, risk-minimizing, and optimizing effects. Appropriate modeling tools, languages, and methods are used for this purpose. Regarding total cost design, systems engineering has a decisive influence like no other discipline. This is especially the case in the early phases of the requirements analysis and the architectural description of the system, which is the reason why there is a constant attempt to minimize the total cost along the life cycle of a system by means of additional improvement measures, like application of architecture frameworks and reference architectures, during these phases. Based on the described fundamentals, the following chapter considers the state of the art, the subdiscipline, and the creation of architecture descriptions.

3 Architectures

While the previous chapter defined the general terminology of the central concept of a system, its life cycle, and the topic of systems engineering, this chapter focuses on architectures and their creation as a sub-discipline of systems engineering [27]. A distinction is made between the two forms of architectures, namely of system architectures and reference architectures, which are introduced and illustrated in the following. The chapter concludes with considering the architecture framework concept, which forms the basis for an effective and efficient definition of such architectures. An overview of the chapter content is shown in Figure 18.

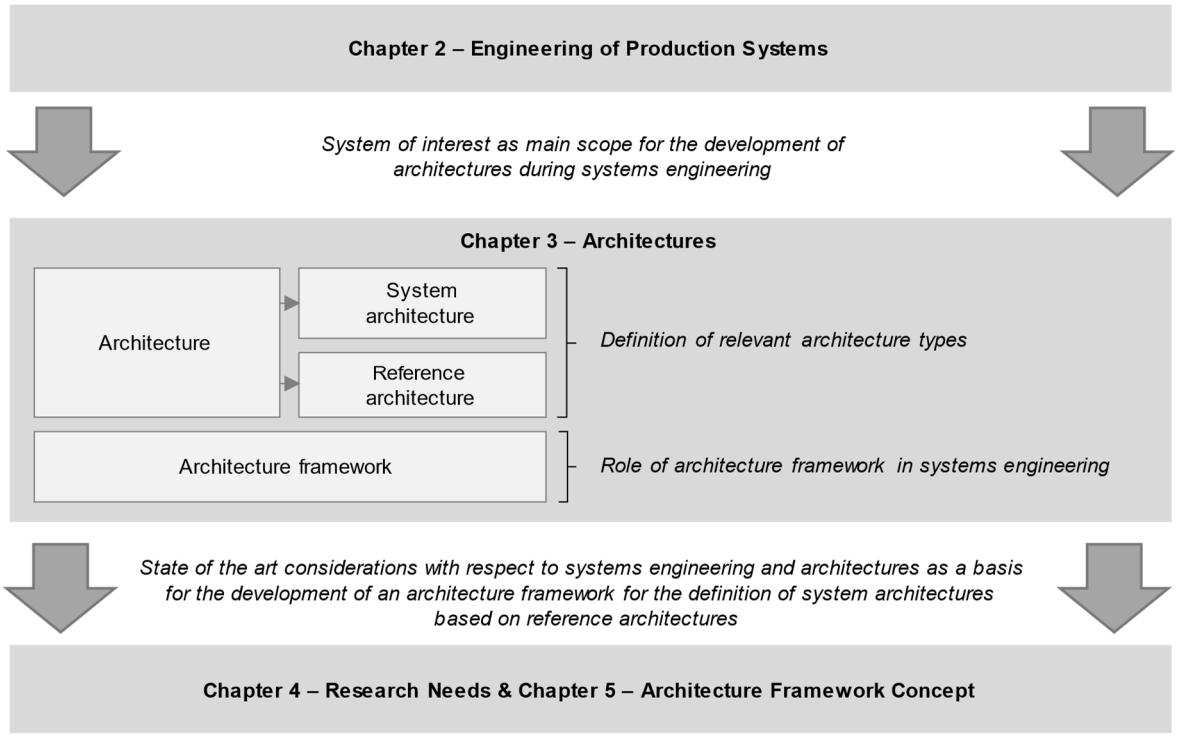


Figure 18: Overview of the contents of chapter 3

3.1 Architecture & Architecting

In several domains the term architecture is defined differently and various interpretations of the term are widely used [27, 61]. Depending on the utilization of the term in the field of interest, architecture can represent "[...] a structure, a process, or a profession" [73]. Within this thesis the term architecture focuses on structural aspects [73], on the principles driving the structure [61], and on the interfaces among the structural objects [118]. Such principles might regard organization and related characteristics of a system [61]. A very general definition can be found in [119] in

which an architecture embodies and describes “the complex or carefully designed structure of something” [119]. Following the argumentation by [61], the given definition is applicable but is missing the related principles. Therefore, within this thesis the following definition will be used.

Architecture

Architecture is defined as “(system) fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution” [30].

The beginning of an architecture's existence occurs with the initial description of the design and the stakeholders [61]. Such an architecture can be of different types, which, among others, are defined by their topic of interest (e.g., operational, security), their purpose (e.g., definition of problem domain, integration), or their focus on a specific group of systems (e.g., system of systems) [27]. An architecture may serve a main purpose and several underlying other purposes [27]. Such purposes can be, for example, the re-use of system elements, context definition, design guidance, and complexity handling [27]. The success and quality of an architecture strongly depends on how well the defined goals and purposes are fulfilled [27, 92]. In principle, the quality of a “good” architecture can be determined if it, for example, “enables competitive products, ensures compliance with present and future laws and regulations, [and that] it can be operated and serviced efficiently and is sustainable" [92]. Furthermore, aspects such as changeability, scalability, and modular design are becoming increasingly important to fulfill the adaptability of architectures required for the future and thus the successful deployment [92, 120, 121]. These separate topics are not discussed in detail in this thesis and further information can be obtained from the sources mentioned.

3.1.1 Architecting

In comparison to the term architecture, which describes structural aspects, the term architecting represents the associated process [73], being concerned with defining and documenting an architecture [27]. In detail architecting can be defined as follows.

Architecting

"Process of conceiving, defining, expressing, documenting, communicating, certifying proper implementation of, maintaining and improving an architecture throughout a system's life cycle" [30].

Architecting is a holistic approach that covers all phases of the system life cycle and is to be seen in connection with projects as well as organizations and can significantly influence processes in these environments [30]. In this context architecting represents the search for the optimal solution while balancing different interests and needs [28]. The result of the architectural effort is a description of the architecture of the system of interest in its environment and with respect to its specific life cycle [30]. The process of architecting can be described as mostly systematic, but also includes actions like context and trade off understanding, analysis of alternatives, and taking decisions [27]. For the sake of completeness, a distinction should be made between the terms architecting and design, which are often used interchangeably and occur in the same environment. [122] in [15] distinguishes that "[architecting] defines what to design, while design defines what to build". If one focuses on architecting and corresponding procedures, one will notice that there are different processes in the literature that describe such a procedure [27]. However, the actual procedure used in reality is strongly influenced and shaped by the architecture context and purpose [27]. Figure 19 shows one rather abstract but well applicable procedure. In summary, an architecture description for a system of interest is created from architecture options that are defined, analyzed, and selected in an iterative process.

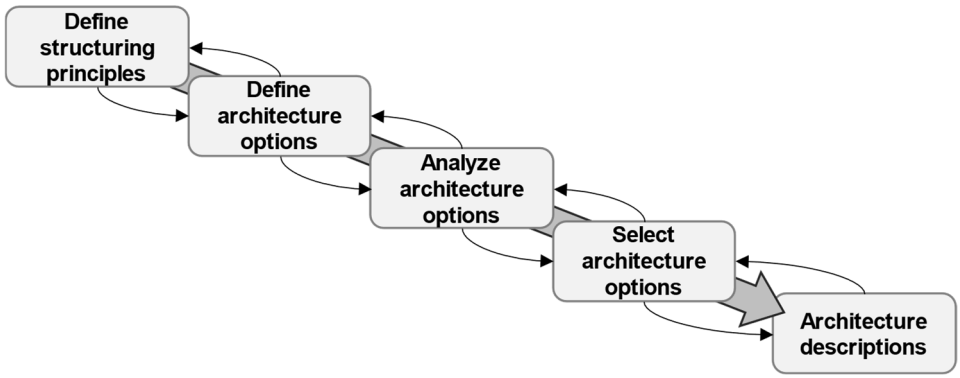


Figure 19: Architecting process adapted from [27]

Finally, it should be mentioned that, like systems engineering, the sub-discipline of architecting must be accepted and implemented in the organization to achieve an efficient and effective use. For this thesis, it is assumed that all architecture related

topics are accepted and supported by the organization and its stakeholders. Details on this topic, such as the structural design of the organization with respect to architecting or the benefits to be achieved, can be found in [61].

3.1.1.1 Architecture Description Utilization

A considered system is reflected in the architecture description, which contains additional detailed information based on the individual needs of the involved stakeholders and the requirements of a project or organization [30]. In this thesis an architecture description is defined as follows.

Architecture Description

An architecture description can be defined as “work product used to express an architecture” [30] of a system.

The architectural description, in contrast to the pure system description, not only maps the form of the system but also reasons why a system of interest has a specific form [61]. Thus, the architectural description also presents which stakeholder concerns, needs, and potential threats were or were not addressed, and the corresponding reasons resolved in the taken decisions [61]. In the architecture description architecturally relevant elements of the system of interest are identified, specified, and documented [26]. Since not all elements are defined in detail in connection with the creation of the architecture description, some elements that are relevant, for example, for the design of the architecture must be added during this phase [26]. In general, the purpose of an architecture description “[...] is to explain the architecture of a system to its stakeholders” [61]. Depending on the specific application example, architecture descriptions always have one main purpose and usually many additional secondary ones [27]. Architecture descriptions are utilized by different stakeholders, for example, for

- establishing context, a starting point for system design as well as development activities, and evaluation of alternative implementations [27, 30].
- documenting key technical and business aspects of a system, among others, specific features, taken architecture decisions and their impact [27, 30].
- communication among stakeholders involved in relevant activities along the systems life cycle [27, 30, 61].
- managing and dealing with complexity as well as uncertainty [27].

- fostering reuse of previously defined and available system solution for the development of new systems [27].
- managing and planning of transformation of systems (e.g., migration of legacy systems, extension, or enhancement of systems) [27, 30].
- assessment of systems during their lifetime [30].

Additional usages to the above listed can be found at [30].

3.1.1.2 Content of Architecture Description

Among other, the content of architecture descriptions is defined in the standard ISO/IEC/IEEE 42010, in which is stated that the content represents certain characteristics needed to enable the above-described utilization [30]. As a bare minimum an architecture description shall represent the system of interest and the related context, the involved stakeholders, and the addressed concerns, as well as the relevant system elements [61]. In the best case the architecture description should identify the stakeholders and their needs with respect to the architecture of the system, represent the relevant viewpoints, associated views, included models (that represent the needs of the stakeholders), and should provide relationships as well as justifications regarding, for example, inconsistencies between models, made decisions, notations, and methods [30, 61]. The completeness of the architecture description largely depends on the stakeholders involved and their input. Ideally, all stakeholders should be involved in the creation of the description, but this is usually not possible in a commercially driven environment, therefore the description is typically sufficient but not 100% complete [61]. The two key concepts viewpoint and view are defined below. The term model has already been defined in section 2.3.5.

Viewpoint

An architecture viewpoint describes a “work product establishing the conventions for the construction, interpretation and use of architecture views to frame specific system concerns” [30].

This implies, that a viewpoint represents the purpose of one or more views [39] and that the “viewpoint will specify the model kinds to be used in developing an architectural view that depicts how the architecture addresses that concern (or set of concerns)” [26]. Conventions established within the viewpoint can be, for example, standards, procedures, guidelines, and templates [38]. The idea is to address only

specific associated concerns of different views within a viewpoint [39]. Consequently, the viewpoint only includes information that is relevant to the field of interest [39]. The different viewpoints are used to differentiate or separate content thematically, for example, problem and solution space or structural and behavioral content [39]. In addition, the “viewpoint also specifies the ways in which the model(s) should be generated and how the models are used to compose the view” [26]. A viewpoint controls the creation of the related view, provides information about its contents and the form of presentation [61]. A view is defined as follows.

View

An architecture view is a “[…] work product expressing the architecture of a system from the perspective of specific system concerns” [30] and “[…] from a specific viewpoint and with a specific degree of granularity […]” [34] based on [30].

A view always represents a system of interest of the related viewpoint [61]. The view represents what you see and the viewpoint determines where you are looking from [123]. [38] adds that a view embodies ”[...] any architectural representation describing a single architectural structure that consists of one or more related models of that structure” [38]. The overall architecture from a specific perspective of a stakeholder of the system is represented by specific parts of one or more architecture models having been selected by the architect of the system in order to communicate and be understood by all relevant stakeholders as to enable them to approve that their needs and requirements are realized by the system [38, 123]. Further information regarding content of architecture descriptions is provided by [30].

To summarize this section and following suggestions of [73], the term architecture represents a structure, which is described in a process called architecting, creating as a result architecture descriptions. For the consideration of system concerns viewpoints and related views are used [30]. Depending on the definition of systems engineering, architecting can be seen in different ways [27]. Within this thesis the opinion given by [27] is followed and (system) architecting is considered as a process and subset within systems engineering. This logical hierarchy of concepts is detailed in section 2.3.3 in the context of the introduced engineering process. The significance of the architecture as an important core component implies that this relationship is partly relegated to the background. In the context of this work, the architecture represents a part of the overall development of a system of interest and therefore of systems engineering. The

constellation of terms introduced above can be applied to a wide variety of architectures in existence. The different architecture categories and variations, such as enterprise, system, and reference architectures, arise from the specific scope of a problem area/subject considered by the concerned stakeholders [28]. A representative set of architecture categories is given among others by [28]. Within this thesis only the concepts of system architecture and reference architecture are further considered. The differentiation between the two terms is described in the upcoming sections.

3.2 System Architecture

As described by [28], architectures can have different characteristics. A system architecture describes the strong dependency and relation of the architecture to a specific system [61], with its, in section 2.1 presented, specific characteristics. The generally valid terms presented in the previous section also apply to the concept of system architecture. The term architecture will be expanded for the use within the system architecture section. Typically, the system architecture addresses the context dependent structure and related/ensembled components, the internal collaboration among each other as well as the external behaviors and relations to satisfy specified requirements [27, 28, 38, 61]. Furthermore, system architecture rules, principles, and guidelines exist to address development, design, organization as well as evolution of the system in the long run [28, 43, 61]. Beneath those “[...] strategic decisions, inventions, engineering trade-offs, assumptions, and their associated rationales [...]” [38] are considered. Additionally [61] states that "[each] system has exactly one system architecture and each system architecture belongs to one system". This statement is valid on all different levels, for example, a considered logical system has one logical system architecture and vice versa [61]. In sum, these descriptions lead to the following summarized definition for the term system architecture.

System Architecture

“System Architecture is the organization of the system components, their relations to each other, and to the environment, and the principles guiding its design and evolution” [45].

The system architecture is allocated to the system life cycle and design phases explained in Figure 15. The system architecture and its development have a success-

critical significance in the context of system development and in relation to the total life cycle cost development (see sections 2.3.3 and 2.3.4). Furthermore, it should be noted that the architecture of a system of interest is always present, but can only be made usable through appropriate formalization, for example in the context of an architecture creation process. By formalizing the architecture, possible advantages can then be achieved, for example, in the application of the architecture descriptions created (see section 3.1.1.1).

3.2.1 Creation of System Architectures

In principle, a distinction can be made between a structured/methodical and an unstructured approach when creating an architecture. Unstructured means that no methodical approach established in literature, standards or practice is used. An individual approach can also follow a structure or methodology. However, since these are very individual, usually not documented well, and therefore more difficult to describe or reproduce, such procedures are not considered in the following. In addition, it should be mentioned that a distinction should be made with regard to the definition of system architectures between the creation of the actual architecture and all other system design activities. System architecting is rather abstract, concerned with the overall concept, the system mission, as well as the structure of the system and the system elements [26]. System design is concerned, among other things, with the physical design in detail, the construction, and the implementation of the system [26]. Within this thesis and the section only the creation of the architecture is considered, since the focus is on the specification of an architecture framework for the creation of a system architecture description on the basis of a reference architecture description. The implementation of the architecture description and the related design of the system are not examined. Further information about architecture design can be found in literature, for example, in [26]. In order to present the state of the art and a possible procedure for defining a system architecture, the main inputs, outputs, and relevant roles that influence the process are presented. In addition, a possible system architecture process is presented in Figure 20 and described below.

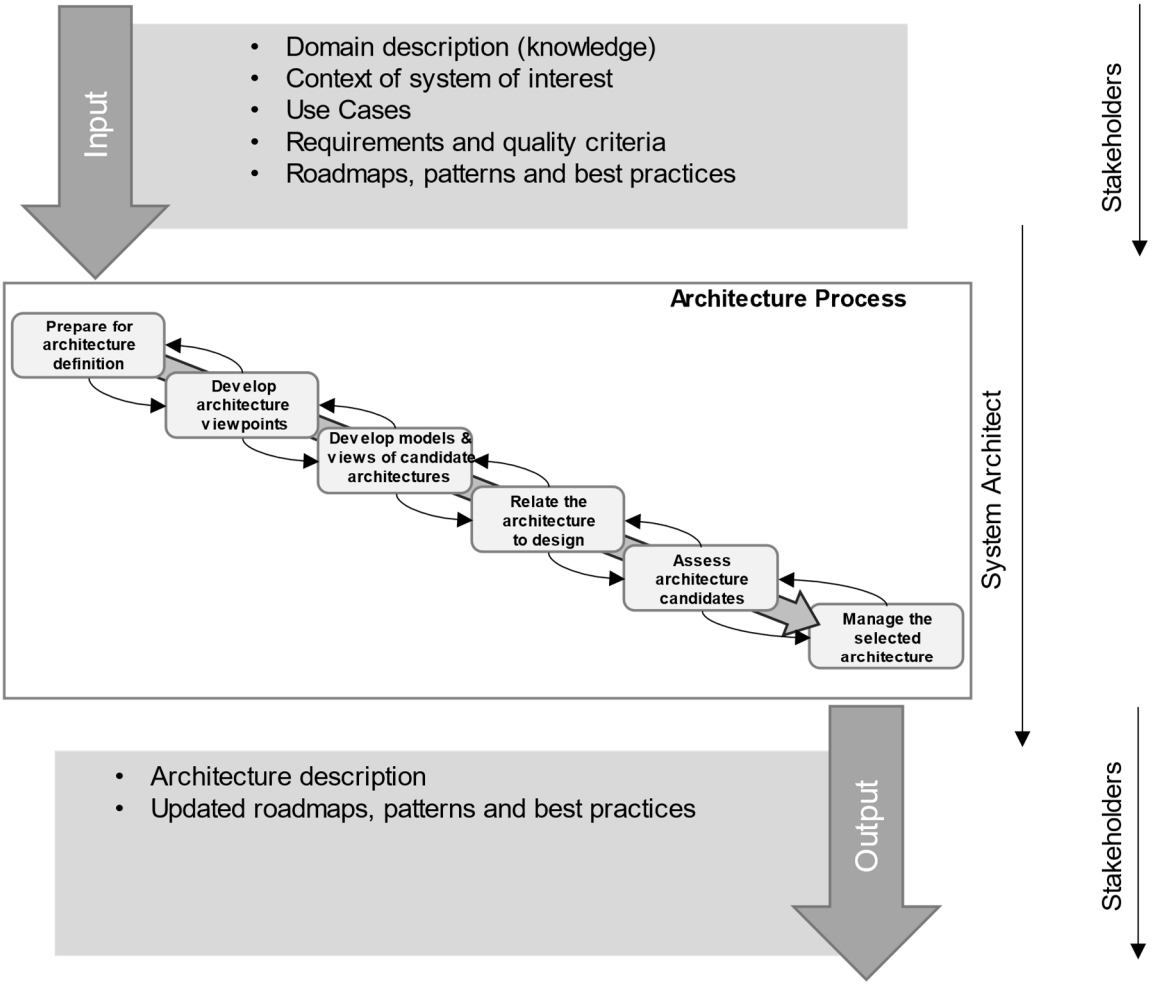


Figure 20: Inputs, Outputs, and Involved Roles of Architecture Process adapted from [26, 27, 61]

The range and selection of an appropriate process depends, among other things, on the application and should be chosen accordingly by the system architect. The creation of an architecture description contains the architecting itself, the evaluation of created content, as well as the approval of the output. The creation itself, can be simplified summarized as definition of requirements and allocation into functions and technical solutions. In addition to the requirements, the knowledge of the domain, the context of the system, possible use cases, qualities, as well as roadmaps, patterns, and best practices are used as inputs for architecting. The knowledge of the domain is elementary for the understanding of the system development as well as a mostly complete definition of the relevant requirements. The context of the system with its interfaces limits the application domain and thus the definition space, for example, for requirements and the architecture (for definition of the context, see section 2.1). Afterwards, on this basis, the corresponding use cases and requirements are defined. Those specify, among others, functionalities or services to be provided by the system. The goal is to design the architecture to be realized in a way that it fulfills the established requirements, which are concerned with different aspects of the system

along its life cycle. Ideally, reoccurring concerns applicable to several systems are considered in form of patterns and best practices, which are constantly kept up to date and maintained across a wide range of projects in order to solve challenges in specific business areas. All the different inputs are provided by stakeholders selected by the system architect (for additional information on stakeholders see [61]). The core activity of the system architect is to steer and drive the architecture process. The main output of this activity is the architecture description, as well as some important additional deliverables, such as documentation of experiences and decisions made. These outputs are then consumed by relevant stakeholders and used for further tasks, for example, the implementation of the system of interest (as shown in the V-model in Figure 15). [61]

The architecture process itself uses the inputs to iteratively define varying possible solutions, evaluate them, and select appropriate technical system elements that form the overall system [26]. In addition to the general process already shown in Figure 19, specific system architecture processes are defined depending on the field of application. In the following, the system architecture definition process from the Systems Engineering Handbook [26], which also originates from the International Council on Systems Engineering (INCOSE), is presented as an example. This process makes greater use of the contents of an architecture description defined in ISO/IEC/IEEE 42010 and presented in 3.1.1.2. [26] defines the following steps for the system architecture process:

(1) Prepare for architecture definition,

(2) develop architecture viewpoints,

(3) develop models and views of candidate architectures,

(4) relate the architecture to design,

(5) assess architecture candidates, and

(6) manage the selected architecture.

While preparing an architecture definition, different aspects are important. Basically, to gain an understanding of the models and views to be defined, all relevant information such as markets, stakeholders, and businesses, must be evaluated. In doing so needs and concerns of relevant stakeholders must be identified, and the different requirements must be derived and analyzed. From these requirements qualities and constraints must be separated. Qualities and constraints deal, among other things, with different conditions in the life cycle and have an influence on how

these are considered whilst creating solutions. Therefore, they also have a strong influence on the design of the system. After these preliminaries are completed, an approach has to be defined how the architecture should be characterized and which enablers are needed during the process. After the process step “preparation for architecture definition”, the architecture viewpoints are developed to define the corresponding models and views. For this purpose, the relevant viewpoints, supporting models, and necessary frameworks have to be identified and established. To develop the models and views of candidate architectures, the necessary tools and modeling techniques are selected first. Then, the context, interfaces, and possible interactions between the system itself and context elements are defined. Based on this, requirements and relevant architecture elements, for example, functions and physical elements, are defined and mapped. This is then represented in models that depict the needs and requirements of the stakeholders best. Such models can be, for example, logical and physical models. These models can represent structural as well as behavioral relationships. The views, which ensure that all requirements have been sufficiently taken into account, consist of the created models which should be analyzed, verified, and validated before the next process steps. For this purpose, appropriate modeling or simulation tools can be used. Subsequently, the architectural options are related to the design. For this purpose, the solutions defined so far are brought into a context with system elements and the interfaces are described. Based on the creation of the architecture options and the defined relationships, a preferred architecture is then created. The selection is made, for example, on the basis of a system analysis or under consideration of a risk assessment. The final step in the process is to manage the selected architecture and to maintain and update it accordingly. [26]

The described process represents one possibility among many for creating an architecture description. A similar procedure is, for example, described in [61]. It should also be mentioned that such an architecture process utilizes on several related procedures, such as the requirements engineering process [61]. In general, it can be stated, when considering the general process of the previous section, that for the creation of a (system) architecture description, inputs and preliminary work are created, architectural options are defined, analyzed, selected, and combined into an architectural solution, taking requirements and stakeholder concerns into account along the whole process.

As an example of a production system architecture, reference is made to the system architecture derived in chapter 7 of this thesis, being based on the introduced application example (see section 1.4). Since the application example is considered in

more detail in the context of the evaluation, the reader can get a good impression of a possible system architecture of a production system. Thus, there is no need to introduce the reader to an additional example of a system detached from this thesis. If there is a need for a representation of a system architecture beyond the example presented in this thesis, another illustrative example defined in a model-based environment is presented in [28], which, however, originates from the field of aviation.

Influence of Agile Approaches on Systems Architecting

For the sake of completeness, it should be mentioned at this point that, due to the agile efforts in systems engineering, agile influences can also be determined in the subarea of the creation of system architectures, leading to challenges for the system architect [61]. Few complete agile systems engineering approaches are known, so far, due to the difficulty of implementing necessary changes [61]. The partial implementation of agile approaches in an organization can lead to a mix of stakeholders who follow different agile and traditional engineering approaches, which holds the challenge for the system architect to coordinate and bring together the different starting points of the approaches followed in order to create a specific system architecture [61]. Since these challenges are of a rather organizational nature and do not directly affect the actual process of creation, but rather the inputs, the topic will not be considered further in this thesis.

3.3 Reference Architecture

In general, the later utilization of the reference architecture defines how a reference architecture has to be designed and thus also how the reference architecture can be used in the architecture creation process [28]. Due to the existence of overlapping and conflicting definitions, the term reference architecture is not clearly defined in literature [28]. Conflicts often arise when the focus of a potential reference architecture is falsely on the procedure of creating an architecture, instead of on the representation of a concrete architecture of a group of systems of a domain [28]. In the context of this thesis, the procedure of creating architecture content clearly falls into the domain of architecture frameworks, which are considered in the subsequent section 3.4. Due to the linguistic labeling and the actual contents, which sometimes blur, a separation is usually not to be made clearly and can be steered argumentatively in both directions. Within this thesis the term reference architecture is defined as follows.

Reference Architecture (RA)

A reference architecture is defined as "[...] a reusable architectural vision for use on systems within a product line or application domain" [38].

In contrast to the system architecture, the concept of a reference architecture always refers to a group of systems. This means that the reference architecture, as described in the definition, can be reused in the form of a template for the definition of architectures for all systems of a class or domain that are in the scope of the reference architecture [124]. The reference architecture represents the characteristics of the considered systems in the form of functional, logical, and physical aspects [28]. In addition, it is further described that most reference architectures mainly put their focus, besides the required requirements consideration, on the functional and logical consideration as well as on frequently used physical details [28]. A specific technical architecture description is often omitted, as this might lead to the reference architecture losing its validity for the considered group of systems. The goal is to provide support for the definition of system architecture in the form of a functional and logical design, which can then be customized, taking into account the specific stakeholder requirements of a system of interest [28, 125]. Due to its intended use, the reference architecture is less concrete than a specific system architecture of the same group of systems or domain [28]. To better understand the connection between a reference and a system architecture, the level of abstraction and the concept of system groups, which are the main reason why this ratio is reflected in this way, are explained in section 0. The content coverage of a reference architecture therefore depends strongly on the scope of the group of systems of interest and the degree of abstraction. Similar to the system architecture, a reference architecture also has different levels of granularity, i.e., levels of detail (for the definition of granularity, see section 2.1).

In principle, such an architecture can be created as shown in the system architecture section but taking into account the input from several systems. An alternative option is the application of an "[...] architectural framework to a class of systems to provide guidance and to identify, analyze and resolve common, important architecture concerns [...]" [124] whose outcome is a reference architecture. The creation of such a reference architecture as a basis for the development of specific system solutions is described in section 3.3.2.

3.3.1 Utilization of Reference Architectures

Unlike a system architecture, which every system possesses, a reference architecture acts in a different role that is not necessarily of relevance for the definition of a specific system architecture. A reference architecture can be of value, for example, "[...] in environments with a high multiplicity factor, creating social, organizational, business, application and technical complexity" [29]. If the use of reference architectures is considered to be advantageous in certain application scenarios, like any other new methodology, process, or procedure that is to be used effectively, the reference architecture must be introduced as a concept into the organization and anchored in the systems engineering strategy and the associated processes [28]. After a successful holistic implementation, the benefits such as reuse and guidance can be leveraged during the architecture development process and will ultimately lead to an increase in speed to market, competitive pricing, and reduction of testing efforts [28]. As mentioned in the systems engineering chapter, organizational topics such as the successful implementation of a reference architecture in the systems engineering process are not considered in this thesis. Rather, it is assumed that a reference architecture has already been implemented successfully and might have following advantages:

- Reduction of development effort and risk in the early development phase of a system architecture, through predefined architecture content such as, for example, requirements and their structure. The effort required to create and maintain the reference architecture can, among other things, be cashed in over several projects by avoiding undesirable developments and by reducing the additional effort required for a definition from scratch. [28]
- Increase in effectiveness during creation by, e.g., using synergy, best practices, and architecture templates. The reference architecture helps to identify possible synergies and assesses in advance whether an increase in efficiency can be achieved. The use of previous experience documented in the form of best practices can also support the definition process effectiveness [29].
- Creating a consistent architecture design, as well as better interoperability and cooperation between systems, by using a reference architecture for the definition of several systems of a domain. Since the use of a reference architecture for the design of the systems is based on a common foundation and the systems thus have, among other things, similar design philosophies, interfaces, and functions, these characteristics can be achieved to a greater extent. [28]

- Reuse of single components across the system is driven by concepts like modularity and other open architecture principles. The reuse approach can be strengthened even further by the application of the reference architecture and its, for example, structuring, uniform definition of interfaces and functions, as well as common data models. Due to the reference architecture approach, some elements of a system are already known by the developers, opening up potential opportunities in the direction of strategic vendor relationships and possibly influencing development and pricing. [28]
- Improves communication through common classification (taxonomy) and enables reduction of efforts based on a unified architecture vision and the possibility of modularization. This allows efforts to be structured better and integrated with each other at a later point in time [29].
- Increasing architecture and ultimately systems engineering quality by using common fundamentals, such as predefined quality attributes as well as the use of proven integration and testing methods, which are provided among others by the reference architecture [28].
- Comply with overall design standards, which can be, for example, represented by reference architectures or architecture frameworks and are being demanded by more and more companies and organizations. Already defined contents, as in refence architectures focusing on such standards, reduce the effort and risk of demonstrating that the content complies with the standards. [28]

The creation of a reference architecture may initially involve higher efforts and initial investments and therefore will not be carried out everywhere. The method offers great advantages for applications where architecture definitions of systems of a specific group or domain are carried out on an ongoing basis and the reference architecture can pay for itself cumulatively through savings achieved over several projects or system definitions [28].

3.3.2 Creation of Reference Architectures

For the definition of a reference architecture certain inputs are required and, as an absolute minimum, a definition process that generates the corresponding results. Additionally, it should be ensured that the created reference architecture contents can be reused for example in form of a repository or integrating reference architecture model [28]. Depending on the abstraction level, the reference architecture is defined with content that may vary between very general and more specific. For example, a

reference architecture of a certain category or domain represents more abstract content than a reference architecture of a certain specific product line [28]. The degree of abstraction of the reference architecture depends strongly on the planned application and the considered group of systems.

The reference architecture is based on inputs of operationally effective systems, system designs as well as related feedback/lessons learned and is primarily used for design reuse in the reference architecture-specific domain [28]. In addition, as possible input for the definition of a reference architecture, a reference model, and information on customer policies, standards, as well as possible new and upcoming products and technologies might be of relevance [28]. The reference model as input represents a basic model, which, among other things, deals with rules and policies applicable for the definition process at hand and can be used to derive models for specific purposes, for example, for the creation of a reference architecture [28, 111]. It is important to distinguish whether the reference model can be used as possible input for the reference architecture process or if the reference architecture model represents an output of the process (for definition of the term model see section 2.3.5). Among others, customer policies and standards are considered as possible input, because it is important to describe the operational scope of the systems based on the reference architecture sufficiently, in order to ensure their operational functionality [28]. Similar factors apply to the inclusion of existing and possible new products as well as technologies in order to derive systems that represent the current state of the art and, ideally, can take future developments into account and integrate them [28].

The actual reference architecture process is similar to the processes described in section 3.1.1 in Figure 19 as well as in section 3.2.1 in Figure 20 and also considers requirements, reference architecture modeling, and validation of the results. The main differences are primarily related to the input, like the consideration of a group of systems to a single system, and the further use of the reference architecture process output for the derivation of system architectures. For this reason, another exemplary process is not presented again at this point, but rather the relationship between reference and system architecture as well as prerequisite for the use of reference architecture content in the transition to a system architecture is examined in more detail. In order to exploit the full potential of a reference architecture for its intended purpose of serving as a blueprint for the definition of a system architecture, the stakeholders concerned must be able to easily identify and use the contents created [28]. As already mentioned in the introduction, this can be done by using a repository or integrating model in which the model content as well as other artifacts and additional important information are documented [28]. One possibility to realize this is

the use of modeling tools, as described in section 6.1, which can be considered beneficial with respect to complex systems. In such tools, models with associated additional files and documentation can be created, searched, and validated [28]. Making this information available within the affected organization is an integral part of the development of a reference architecture description, as well as the creation of this information [28]. With regard to the relationship between the reference and the system architecture, it is also important that the information flow is not unidirectional and that there is a constant mutual exchange of information between reference and system architecture so that users are able to make changes and extensions [28]. However, this should only be possible in a uniformly regulated procedure, as unstructured changes can quickly lead to inconsistencies in the reference architecture and make it unusable [28]. How governance for quality assurance and maintenance of architectures can be structured and implemented is described in [28] and will not be considered further in this thesis due to its organizational character. Nevertheless, this is an eminently important aspect for the successful use of reference architectures. The advantages arise when a new system is to be developed within an organization, and the stakeholders concerned can rely on the predefined contents, which they transfer into a specific system description with the aid of a methodical procedure like incorporated within an architecture framework [28]. The reference architecture does not make such a definition method redundant but supports it by reducing the initial effort that would otherwise be required in the early phases of the architecture process [28]. With regard to the possible content scope of a reference architecture, it should be noted that it is usually easier to eliminate superfluous content than to redefine and integrate missing content [28]. For this reason, the goal when creating a reference architecture should always be to define as comprehensively as possible all alternatives that could be relevant for the system group or domain of interest [28]. After the relevant contents of the reference architecture have been selected for the definition of the system architecture, they will most likely have to be adapted, for example, with regard to interfaces and user roles, so that they meet the specific requirements [28]. All in all, with a good reference architecture, the creation should generate less effort than with a scratch definition of a new system architecture [28]. Plus, the effectiveness and efficiency of using a reference architecture depends heavily on the methodology and tools used holistically throughout the organization [28].

3.3.3 Reference Architectures for Production Systems in Literature

Due to the different understandings and definitions of reference architectures and associated models the state of the art in terms of reference architectures is difficult

to outline. In order to give an impression of what a reference architecture in the area of production systems might look like and to present the concept in more concrete terms, in the following widespread but more abstract approaches as well as a probably moderately known but more concrete reference architecture are presented. Finally, a conclusion with respect to the thesis is drawn about the considered reference architectures.

Similar to the results with respect to architecture frameworks (see section 3.4.5), the literature research conducted in [126] shows that established and widespread reference architectures or reference architecture models are mainly abstract and general in form and content. It is assumed that this might be caused by the higher applicability due to the more general focus or because the definition of specific reference architectures becomes more difficult with the shift from abstract to concrete. A reason behind this might be that more concrete contents require more architectural decisions, which will result in the scenario that the application area becomes more and more limited. This is one of the reasons why reference architectures often have their focus in the functional/logical area [28].

In summary it can be stated, that the investigated reference architectures Industrial Internet of Things Reference Architecture (IIRA) [124, 125], Internet of Things Reference Architecture (IoT-RA) [127], and Reference Architecture Model Industry 4.0 (RAMI 4.0) [128–130] represent valid approaches on an abstract level, which are applicable in different domains, as for example in the domain of production systems [126]. However, it also became clear during the research that the reference architectures cannot be used directly as a template for the derivation of specific production systems without relatively large additional effort [126] in order to bridge the big gap in abstraction, which negatively impacts the utilization of a reference architecture and the creation of a related system architecture [89]. Therefore, as a more specific example the reference architecture created within the research project CrESt is introduced in the following.

Reference Architecture Example from Research Project CrESt

In the following the reference architecture form the research project CrESt will be presented, which is later used for the conducted prototypical application of the architecture framework concept. Within the research project a model-based reference architecture with associated content was defined for the domain of adaptable and flexible factories, which thematically includes the domain of production systems [35]. The development methodology presented therein, as well as the application example

of the reference architecture for the adaptable and flexible factory, is based on the core topic of the research project CrESt; collaborative embedded systems (CESs) [35]. These systems interact with other systems in a collaborative system group (CSG) to achieve a common goal [35]. The systems are flexibly coupled and can be adapted accordingly when changes are necessary [35], as it is required in an adaptable and flexible factory for production changeovers. The general development methodology is based on the SPES_XT modeling framework also presented in this thesis in section 3.4.5, which in short considers requirements, functions, logical components, and technical solutions [35]. Further information on the modeling framework can be looked up in the mentioned section of the thesis or, among others, in [33] and [34]. According to [35] in a first step, a reference architecture CESs and CSGs was defined based on the modeling framework. Afterwards, based on the subsequent results and in combination with the methodology, a specific reference architecture for the domain of the adaptable and flexible factory was created [35]. The focus of the reference architecture is on selected core requirements and, above all, on the creation of a functional and logical reference architecture [35]. In order to remain as independent as possible of specific technical solutions, no technical reference architecture was defined [35]. According to [35], the specific requirements were mainly taken from the application scenarios of the Platform Industry 4.0 [66, 67], as well as the use cases that are described in [35]. The goals for the systems of the adaptable and flexible factory that result from those considerations are first and foremost regarding the production process as well as the production of the products [35]. In addition, related goals arise, such as reconfigurability of the production system, analysis of product orders, planning of production and capacities, optimization, maintenance, collaboration, and evolution of the product portfolio [35]. An exemplary excerpt of the logical reference architecture for the adaptable and flexible factory described in [35] and derived from the formulated goals and the general reference architecture is shown in Figure 21. This exemplary excerpt displays the CSGs that are relevant from a logical perspective at the highest granularity level of the reference architecture with respect to the adaptable and flexible factory. On subordinated higher granularity layers the shown contents are detailed and address the concepts relating to roles, functions and goals described in general form in [35].

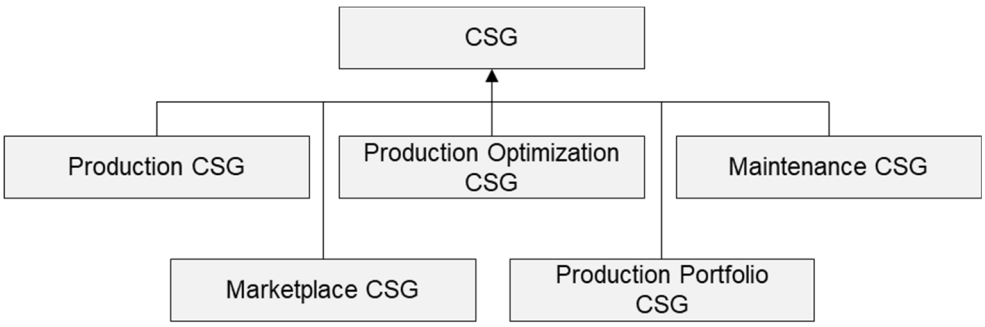


Figure 21: Example of high-level logical reference architecture for the adaptable and flexible factory [35]

In general, CSGs reflect different goals and functions they are intended to fulfill [35]. The ProductionCSG is representative of a system group that is concerned with producing a product based on an order [35]. For this purpose, different functions, such as the analysis and maintenance of the production order, the monitoring of the production, or the collection and provision of data for operational use, are taken into account [35]. It should also be noted that when considering logical elements, no statement is made about which specific system will fulfill which specific task or function in the later technical solution. If, for example, in the context of the ProductionCSG, different functions are mentioned, which are unifying represented by a single symbol, this does not mean that in the technical solution all these functions are fulfilled by only one system but rather by the represented group of systems. The ProductionOptimizationCSG deals with the improvement of the production and thus, among other things, with the support of the operator in the form of bottleneck detection, failure support or capacity optimization of production [35]. The MaintenanceCSG considers all functions and tasks that support the optimal operation of the production system/factory [35]. These functions relate, for example, to the planning, execution, and prediction of necessary maintenance tasks [35]. The MarketplaceCSG deals with collaboration topics, especially between factories and at a subordinate granularity level between systems and serves as a hub for offering and utilizing required capabilities [35]. The ProductPortfolioCSG is concerned with the continuous improvement of the factory as well as the production and therefore includes all functions considered with, among others, the analysis of product orders or the specification and proposal of improvement measures [35].

On a detailed level, the CSG is formed by different CESs, such as the ProductionCES shown in Figure 22. As shown and depending on the application scenario a ProductionCES might have a certain role within a CSG, a ProductionCapability, and will utilize several mangers, for example, for managing the production itself or optimization.

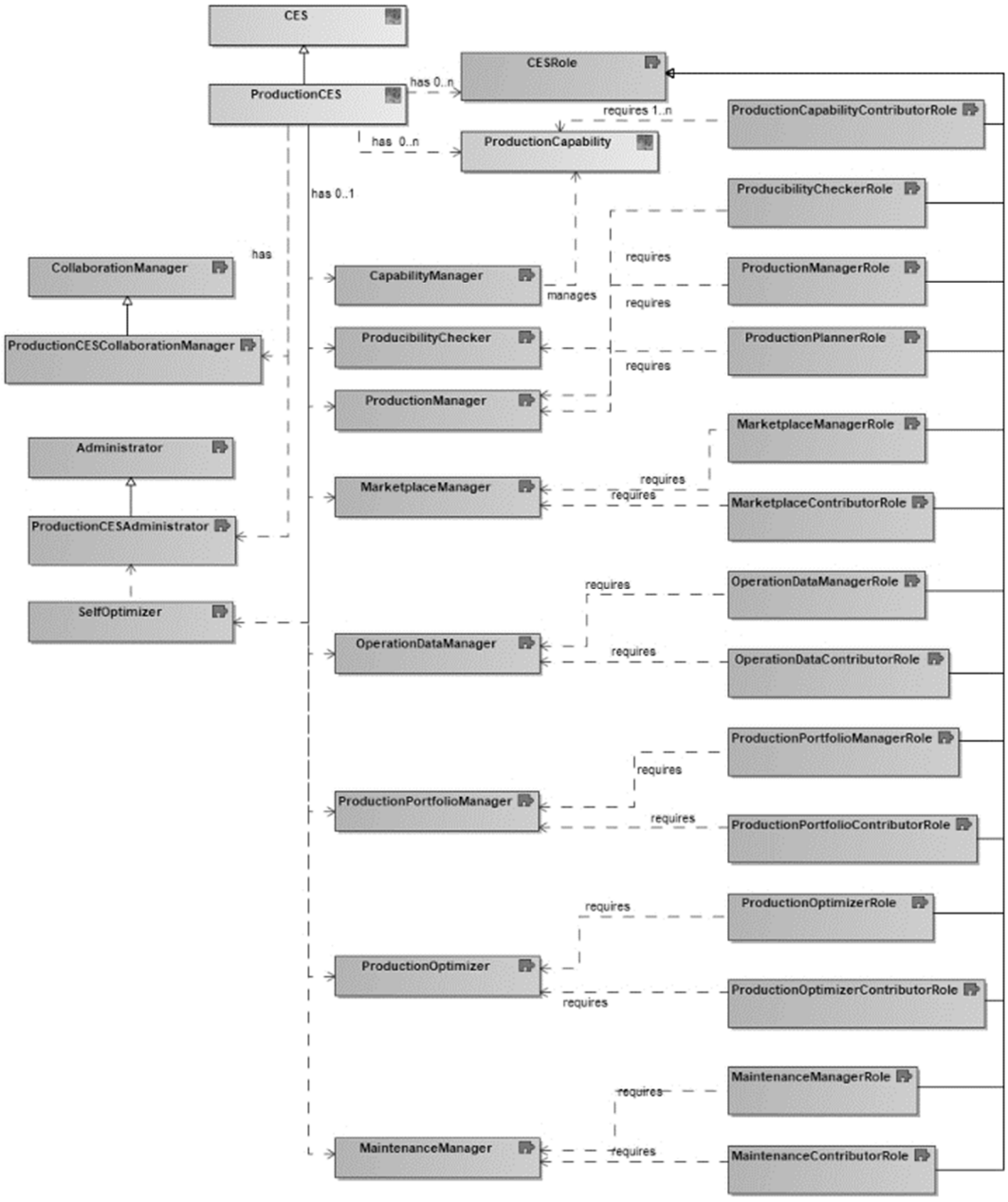


Figure 22: Exemplary detailing of ProductionCES of reference architecture [131]

These contents described above were detailed further with respect to the different CSGs and CESs of the reference architecture of the adaptable and flexible factory but are not further elaborated in the form of image or text by the author of this thesis since the reference architecture and related contents were developed by Siemens AG in the context of the research project CrESt and beyond and have not yet been published completely. Nevertheless, the model-based reference architecture described above can be used for the later conducted application of the specified architecture framework.

3.4 Architecture Framework

Considering the concept of architecture framework there are different conceptualizations, which lead to the fact that the term architecture framework is regularly misinterpreted as a template or basic architecture [61]. However, as described in the previous section, such basic architectures with template character are referred to as reference architectures, posing rather complementary components of the architecture frameworks and are ideally defined by using a selected architecture framework [28, 61]. This already outlines the actual characteristics of the architecture framework, which, in simple terms, is to provide an approach to develop, structure, organize, and document architectural description in order to better support the architect by creating system or reference architectures [27, 28, 61, 132]. Within the design phase the architecture framework represents one main concept for system architecting and the creation of system and reference architecture descriptions. In the following, the term architecture framework as a key concept within this thesis, the utilization of architecture frameworks, advantages and disadvantages, as well as the key components of an architecture framework are defined. Concluding, a representation of widely used and recognized architecture frameworks is analyzed with respect to specific requirements framing the assumed research gap concerned within the thesis.

3.4.1 Architecture Framework Term Definition

Nowadays it can be stated that the support of architects during the creation of a system architecture description primarily takes place in the form of architecture frameworks [15, 27]. The application of these architecture frameworks mainly serves to structure and organize the architecture descriptions of the considered system as well as to standardize development efforts and related systems [15, 27, 133]. In standard ISO/IEC/IEEE 42010 the term architecture framework is defined as follows.

Architecture Framework

An architecture framework represents ”[...] conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders” [30].

This use of pre-defined structural procedures, products, and design principles, which are described in the context of the architecture framework for the development of

system architectures, are detailed in the following [15]. ISO/IEC/IEEE 42010 also states that the architecture framework not only specifies procedures for the creation, interpretation, and analysis of architecture descriptions, but also for the use of these descriptions [30]. Furthermore, it is described that architecture frameworks, in addition to the creation of architecture descriptions, are also used for "[...] developing architecture modeling tools and architecting methods; and establishing processes to facilitate communication, commitments and interoperation across multiple projects and/or organizations" [30]. It is further pointed out that the components of an architecture framework should include information for identification of the framework, consideration of stakeholders and their concerns, viewpoints that take these concerns into account, and associated rules [26–28, 30, 134].

As in the process of creating an architecture description the identification of the stakeholders, who have fundamental concerns for the architecture, is also relevant as a starting point within the architecture framework [30]. An overview of relevant stakeholders, such as developers or operators of the system, is presented in [30]. Possible key concerns that should be investigated in this context include the purpose of the system, the use of the system, and risks in the life cycle of the system [30]. Other concerns are also described in [30]. The terms viewpoint and view (described in section 3.1.1.2) can be defined as part of an architecture description, a framework or individually [30]. A viewpoint should be used to put the identified stakeholder concerns into an architectural context [30]. In addition, the viewpoint shall represent which model kinds are used to map these concerns [30]. For each of these model kinds, "[...] languages, notations, conventions, modeling techniques, analytical methods and/or other operations to be used on models of this kind [...]" [30] as well as associated links to sources shall be presented [30]. This can be done, among other things, in the form of a meta-model, which reflects the content that can be described by using the defined viewpoints within an architecture [27, 30]. In addition, the viewpoints should provide information on how the individual views of a corresponding viewpoint, being part of the architecture description, can be created, interpreted or analyzed, for example, by applying appropriate methodologies or templates [26, 30]. Based on those views, among others, analysis on the created architecture can be performed [132]. Furthermore all correspondences shall be considered, which define the relationship between elements within an architecture and thereby expresses architectural relationships in an architecture description [30]. Taking these rules into account, it can be expressed whether an architectural relationship has been fulfilled to the defined extent or for what reason it is violated [30].

From the author's point of view, it is important to mention at this point that the concept of the architecture framework can support the stakeholder(s) in their work and suggest appropriate methodologies, content, and structures. The stakeholders and their capabilities as well as expertise continue to determine the success or failure of the application of a framework and the related outcome. In addition to factors such as acceptance and suitable organizational structures, human capabilities such as intelligence and creativity, decision-making behavior, recognition of dependencies, assessment of importance and urgency, consistency and flexibility, and dealing with errors play a key role with respect to good problem-solving abilities in the development process [135]. These human capabilities can be supported to a certain extent methodologically and in terms of content by the architecture framework but cannot be replaced. Human capabilities should therefore always have a special focus in relation to the use and definition of, among other things, architecture frameworks.

3.4.2 Utilization of Architecture Frameworks

After the general definition of the term architecture framework has been introduced, this section will deal specifically with the use of an architecture framework. As described within ISO 42010, the architectural framework represents "conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders" [30]. Depending on whether the application of the framework aims at a single system or a group of systems in the domain of interest, the results differ. In case of an application to a considered group of systems, the subsequent result describes a reference architecture [124], which considers and describes the entirety of all systems of the group. When applying the architectural framework to a single system, the result is a system architecture valid for the single system of interest. This fundamental distinction and the corresponding result are shown in Figure 23. Additionally, S.W. Lin et al. describe that a reference architecture can be used as a template for the definition of individual systems belonging to the group of systems for which the reference architecture has been defined [124] (also see Figure 23). This transition between reference architecture and system architecture and the necessary support by an architecture framework describes the research focus of this thesis and is elaborated in detail in the following.

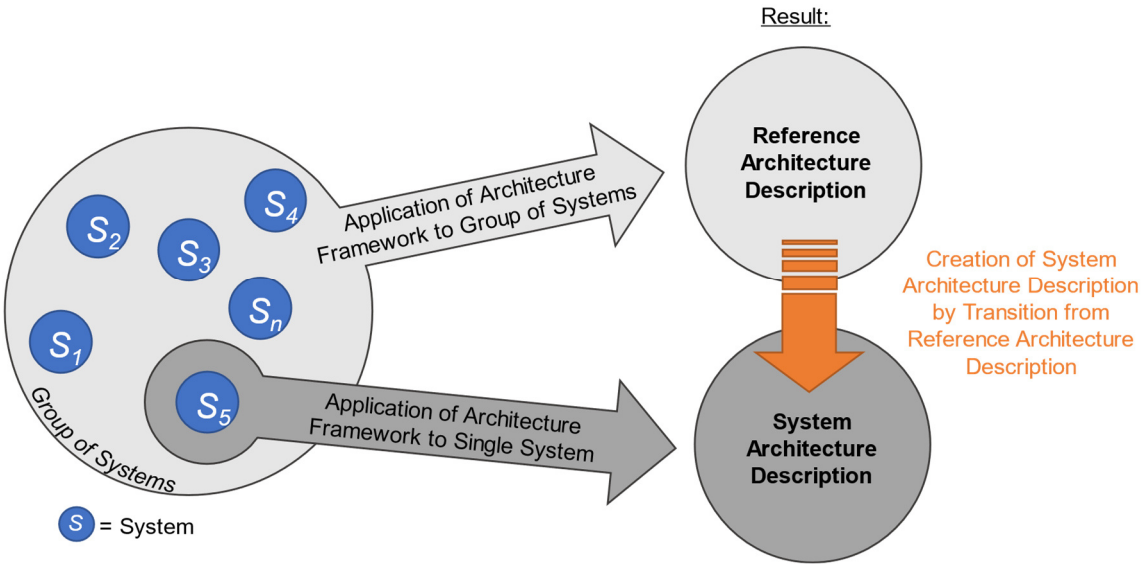


Figure 23: Application of Architecture Framework and Results

In principle, the statement can be made that for the definition of architectural content either (1) a structured methodical approach, (2) a free procedure not guided by written down methodologies or (3) a mixture of both can be applied. The structured methodical approach describes the application of an architecture framework, which represents the application of clearly defined contents for the guidance of architectural definition. The second and third approach represent a possible working method of a stakeholder, who only partially falls back on clearly defined and documented methodical procedures and combines or executes individual definition steps by implicit expert knowledge. This does not mean that the stakeholder carries out the definition without a clear procedure or methodological principle, but rather that exclusive knowledge or methods are used not available, understandable, or reproducible by third parties. Such procedures and methods might have been devolved and evolved over time. This makes it difficult or impossible for third parties to comprehend, who are, for example, supposed to implement the system on the basis of the created results. A definition of a system of interest or a group of systems of interest is possible in this way, but because of the lack of traceability and relationship between elements and the lack of knowledge on utilized methods and concepts those procedures are more error-prone and much more complex for third parties to understand. Therefore, the application of architecture frameworks is a way to create sustainable and long-term applicable architectural descriptions.

Production System Application Example

To clarify the relationship between reference architecture, system architecture, and architecture framework, it is assumed that a group of systems is considered that belongs to a similar domain or considers related content. This could be, for example, a group of systems that considers the production of a component of an engine. In this example, the production system application example "cylinder head manufacturing" represents one of these specific systems. Others are the "engine block manufacturing system" or "intake manifold manufacturing system". For each of these individual systems, a system architecture description can be defined by individually applying an architecture framework to them. The same applies for the definition of a corresponding reference architecture. However, in this case the totality of all systems is considered, and the architecture framework is applied to the group of systems. The result could be, for example, with a more specific consideration in mind, a reference architecture for engine manufacturing systems or, for a broader consideration and greater applicability, a reference architecture like the CrESt reference architecture for the adaptable and flexible factory, which was presented in section 3.3.3 and is used in this thesis for evaluation purposes. Based on such a reference architecture, predefined elements can then be transformed into architectural descriptions for specific systems of interest, which is part of the group under consideration.

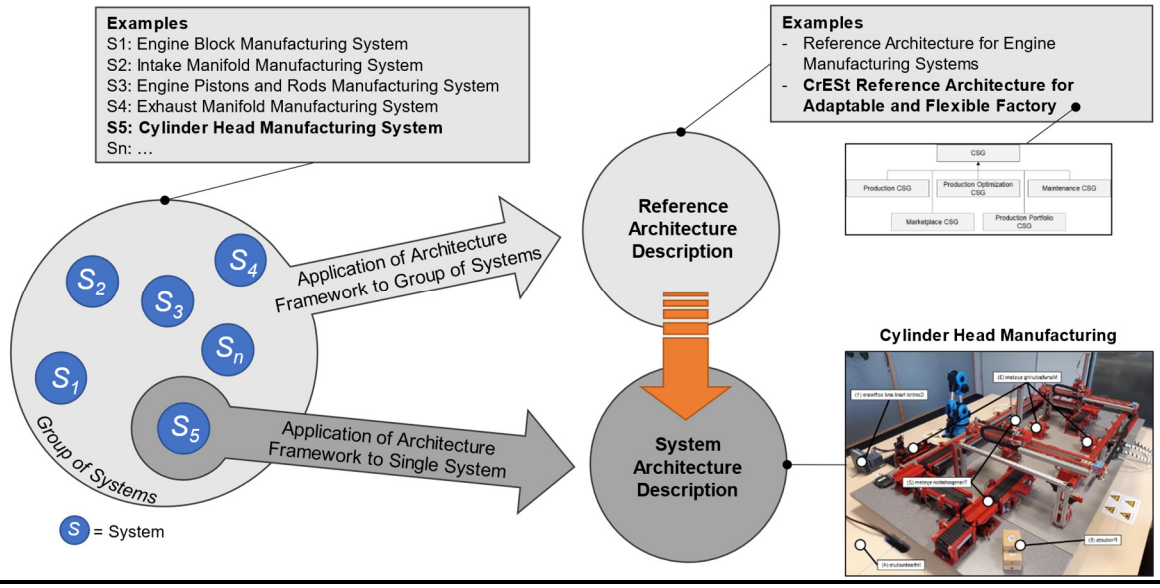


Figure 24: Production system application example – application of architecture framework

3.4.3 Advantages and Disadvantages of Architecture Frameworks

As with the use of a system or reference architecture, there are also certain advantages to be achieved as well as disadvantages that arise when using an architecture framework. In the following, advantages and disadvantages of using an architecture framework are listed, which shall always be considered in a differentiated

manner and dependent on the focus and type of the architecture framework. For example, both a management and a technically oriented framework will support the business purpose of a company/organization, but to a different extent based on the focus and goal of the architecture framework. Some benefits that can be achieved by using different architecture frameworks are, for example, business purpose support, complexity management, decision support, standardization [132] or stakeholder-independent knowledge assurance. The business purpose is supported by all architecture frameworks, especially by enterprise architecture frameworks, and helps the applying organizations to achieve their specific goals [132, 133]. Among other things, enterprise architecture frameworks provide best practices with the purpose of creating usable and economical solutions [132, 133]. In the context of complexity management, a framework provides different aspects and approaches to deal with complexity, including clear responsibilities and structures, having goal-oriented project management, specific tools, and modeling approaches [132]. The different tools and methodologies provided by architecture frameworks also support organizations or stakeholders, among other things, in their decision-making [132] and communication [30]. As a result, corresponding developments and dependencies, for example, with regard to business processes, can be better analyzed, evaluated, and related to each other in order to be able to take decisions [132]. All these methods, structures, and procedures contribute directly or indirectly to the fact that architecture frameworks have an unifying character when applied continuously and contribute to standardization within an organization but also across organizations [132]. This can be relevant, for example, with regard to interoperability [30, 132]. Depending on the framework and application scenario, the use of architectural frameworks can support and positively influence integration and interoperability, holistic consideration of businesses and systems, optimization, data management, security, and stakeholder education [132].

In contrast to the positive aspects, the utilization of architecture frameworks can also pose challenges, such as the initial investment, required experience, and support for implementation of architecture framework throughout the whole organization [132]. Additionally to initial investments and development costs, expenses also include costs for licenses, tools, and knowledge acquired from experts [132]. Due to the complexity and lack of experience, the expert knowledge should be taken into consideration in any case in order to guarantee target-oriented development and use [132]. Besides the cost and knowledge aspects, an essential challenge is to achieve acceptance for the development and implementation at all levels of the organization and by the management [132].

For the decision on whether the application of an architecture framework is advantageous or not, the individual situation with respect to development as well as use of an architecture framework must be analyzed and the pros and cons weighed against each other.

3.4.4 Creation of Architecture Frameworks

The review of systems engineering and architecture literature reveals that no established process has been defined for the creation of an architecture framework. In one of the relevant standards in this field, the ISO/IEC/IEEE 42010-2011 [30], the architecture framework plays a central role, but besides relevant components no clear procedure for the definition of contents is described. In a very similar way and away from standards, [73] uses terms such as goal of the framework, architectural levels, and organization of architecture description to outline the definition of the architecture framework, but without describing a concrete process. It is noticeable that very similar contents are outlined with other terms. Since no clear procedure can be derived from the literature, the following section will focus more on the most important components of the architecture framework. Concludingly, it is described, which components should be contained in the results of the application of the architecture framework in order to highlight how the contents of the architecture framework are projected into the created architecture description.

ISO/IEC/IEEE 42010-2011 aims to propose architectural conventions and established practices, which shall lead to the creation of a common basis for the definition and use of architectural frameworks, in order to, among other things, improve and strengthen the understanding and interoperability between architectural communities [30]. Based on the standard ISO/IEC/IEEE 42010-2011 it shall be exemplarily described, which components an architecture framework should consist of. These components include information describing the architecture framework, identification of concerns and the stakeholders who express those concerns, architectural viewpoints framing those concerns, and relevant correspondence rules [30] and are shown in Figure 25. In addition, it is specified that the viewpoints express which model kinds are used to map the concerns expressed by the stakeholders [30]. The individual components of a framework have already been defined in section 3.4.1 and are therefore not introduced in detail again. The terms and related contents of an architecture framework like viewpoints, model kinds, correspondence rules, stakeholders, and concerns have already been defined in connection with the term architecture description. The relationships can be found in section 3.1.1.2. Considering the stakeholders and associated concerns it becomes clear, that they are mainly considered at the beginning

of the specification of an architecture framework, but ultimately serve to specify the relevant content that is to be represented in the views of the individual viewpoints. Likewise, the correspondence rules describe relations within the architecture description [30], which are represented again in form of views within the specified viewpoints. Therefore, it can be stated that the viewpoint concept and the views specified therein play a central role for the definition of the architecture frameworks.

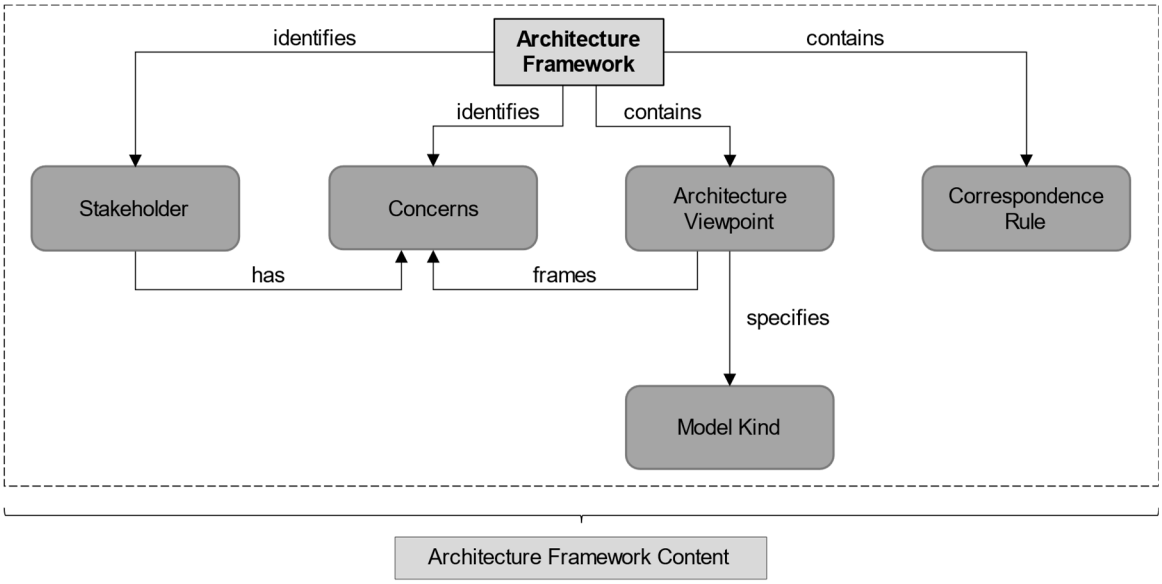


Figure 25: Content of architecture framework (adapted from [30])

As the selection of a suitable approach for the creation of an architecture framework strongly depends on the application scenario and on suitable in literature available (sub-)procedure, for example, for the definition and management of stakeholders [136] or the creation of viewpoints and views [123], this situation is only highlighted here, and no further selection of procedures or details are given.

Based on the components shown in Figure 25 (defined in section 3.4.1), a statement can be made whether the architecture description adheres to the specified architecture framework or not [30]. This is the case if every relevant stakeholder and every related concern of the architecture framework have been identified and taken into account in the architecture description, every viewpoint as well as view and the application of the correspondence rules of the architecture framework are included in the architecture description, and if the architecture description itself conforms to the form described in the standard ISO/IEC/IEEE 42010-2011 and in section 3.1.1.2 [30]. It should also be mentioned, that an architecture description can be assigned to more than one architecture framework [30]. However, it must be ensured that the contents

of the frameworks are compatible and, among other things, coordinated with regard to contents like stakeholders, concerns and viewpoints [30].

3.4.5 Architecture Frameworks in Literature

Since many different architecture frameworks are described in literature, the relevant state of the art is elaborated below, taking into account the topic of this thesis. From the author's point of view, the large number of different architecture frameworks can be attributed to the changing situation over the last years and corresponding adapted developments as well as the development based on specific application scenarios and goals. To be able to perform a targeted assessment of the state of the art by taking into account the specific situation examined in this thesis, corresponding requirements for the assessment are formulated in section 3.4.5.1. These are then used in section 3.4.5.3 to evaluate the architecture frameworks selected and presented in section 3.4.5.2.

The purpose of this literature review is to be able to make a statement about the extent to which the existing architecture framework solutions can or cannot be used for the intended use of reference architecture descriptions in the definition process of specific system architecture solutions. Based on the evaluation, the need for research in relation to this thesis can be derived and specified in detail.

3.4.5.1 Requirements for Evaluation of Architecture Frameworks

There are a variety of architectural frameworks with different focuses and goals considered in literature. To be able to make a suitable statement about the state of the art of architecture frameworks with regard to this thesis, specific requirements for an evaluation are formulated. These requirements should aid the assessment of the applicability of the architecture frameworks considered and to identify possible gaps or research needs. For this purpose, the specific situation, the resulting need, and the associated thesis specific requirements are defined in Table 2. These requirements are defined based on the problem statement and research objective outlined in the introduction and in the state of the art. The later evaluation conducted in section 3.4.5.3 will aim to show to what extent the specific challenges considered in this thesis have already been taken into account in existing architecture framework approaches.

Table 2: Requirements for evaluation of architecture frameworks (adapted from [137])

No.	Situation	Need	Thesis specific requirements
1	Systems and associated life cycle processes experience, among others, a steady increase in complexity, which places special demands on the design process of such a system as well as on the stakeholders.	Identification of means and methods for the development of complex systems and the support of associated stakeholders.	Architecture framework shall **consider the development of architecture descriptions** of complex systems and ideally of production systems.
2	Architecture frameworks are defined for different scenarios and therefore have a main focus of application. The results of applying the architecture framework reflect this specific focus.	Selection and application of the most appropriate architecture framework for the considered application scenario.	Architecture framework shall **consider the creation of technical architecture descriptions** (focus production domain).
3	Due to the increasing complexity of systems and processes as well as the increasing globalization and decentralization in companies, there is a growing need for, among other things, complexity-reducing and integrating processes and measures that support the relevant stakeholders across disciplines and organizations in the specification of economically competitive solutions in the area of systems engineering and, in particular, architecture development.	Reuse of architecture-relevant content during the definition of architectural descriptions.	The architecture framework shall **allow the use of predefined architectural content** for creation of technical architecture descriptions.
4		Use of predefined reference architecture description for defining a description of an architecture of a system of the system group/domain considered in the reference architecture.	The architecture framework shall **consider the transition** of **reference architecture** description content for the specification of **system architecture** descriptions.
5		Support of stakeholders involved in deriving an architectural description from a reference architecture, for example, through specific methods or processes.	The architecture framework shall explicitly **implement specific methodologies, structures, or content related elements** to support stakeholders in the development of specific system architecture descriptions based on available reference architecture description contents.

3.4.5.2 Architecture Framework Examples

As described in the previous sections, there are a variety of architecture frameworks. Among the most frequently mentioned architecture frameworks, for example, in comparative literature, like [132, 138, 139], in standards, like ISO/IEC/IEEE 42010-2011 [30], in publications of large engineering associations, like INCOSE [26, 27], or of governmental organizations, like NASA [43], are for instance the following frameworks (mentioned in alphabetical order):

A) Department of Defense Architecture Framework (DODAF) [140]
B) Federal Enterprise Architectural Framework (FEAF) [141]
C) Generalized Enterprise Reference Architecture and Methodologies (GERAM) [142]
D) Ministry of Defense Architecture Framework (MODAF)[3] [143]
E) Reference Model of Open Distributed Processing (RM-ODP) [144]
F) The Open Group Architecture Framework (TOGAF) [133]
G) The Zachman Framework [145]

Other architecture frameworks also cited rather frequently are mentioned, among others, by [132, 134, 138].

As already indicated, the focus of the architecture frameworks varies due to, among other things, the type of architecture under consideration. The type of an architecture is specified based on the application scenario, the goal or the purpose pursued [27]. For example, the type of architectures considered may be based on the topic, including, operational or security, on the purpose, including, integration or domain definition, or on a specific system, including, system of systems or enterprise systems [27]. This results in a variety of architecture types, which, for example, depending on the content or degree of abstraction, are to be seen in a certain dependency or hierarchy to each other. The common hierarchization into business architecture, information architectures, solution architecture, and technical architectures can be found in comparable form in different sources, among others, in [61] and [138]. Alterative possibilities of the hierarchization are represented, for example, in [132]. Depending on the type of architecture under consideration, the architecture frameworks can be classified analogously and hierarchized, for example, from enterprise architecture framework to technical architecture framework. This means

[3] After creation of the state of the art MODAF has been withdrawn on 15th of January 2021 and was replaced with the NATO Architecture Framework (NAF) V4. The consideration of MODAF as an example for architecture frameworks within this thesis will not be replaced, as this would not impact the general conclusion in any positive or negative fashion.

that dependent on the selected architecture framework a spectrum of a more general holistic or specific detailed scope of architecture consideration is reflected. An enterprise architecture mostly considers the whole spectrum from business to technical domain, whereas. frameworks with a topic focus might only consider specific architectural contents.

When looking at the frequently mentioned and widespread architecture frameworks as well as at the literature mentioned above, it is noticeable that mostly enterprise architecture frameworks following a more holistic scope can be found. The reason for this cannot be clearly deduced from the literature. In the eyes of the author, various factors could have contributed to this situation. One factor, comparable to implementation of reference architectures, might be that a holistic approach is more effective and, based on experience, more promising for the utilization of an architecture framework. Another factor that more generally applicable abstract approaches have become better established and spread than specific frameworks could be the influence of different peer groups utilizing architecture frameworks. Naturally a more holistic general approach might appeal to a bigger audience than one considering a specific application scenario or challenge. A third factor contributing to the situation might be the starting point and drivers of the adoption and implementation of an architecture framework, such as the creation by or with governments and large international organizations well established within research and standardization. All in all, these factors could have contributed to some architecture frameworks gaining acceptance and being mentioned more often in relevant literature than others.

Nevertheless, literature also contains a wide variety of other architecture frameworks that supposedly consider more specific aspects and represent, for example, technical architecture frameworks. These are not as widespread, well documented, and evaluated by the relevant community as the frameworks mentioned above, but should be evaluated by representative examples when considering the state of the art and the specific focus of this thesis. These architectural frameworks, taking into account the thesis content, include, for example, the following frameworks (mentioned in alphabetical order).

H) Model-Based System Architecture Process (MBSAP) [28]
I) Method Framework for Engineering System Architectures (MFESA) [38]
J) SPES_XT modeling framework [34]

3.4.5.3 Architecture Framework Evaluation

Reflecting the complete state of the art for the evaluation is difficult or impossible due to the large number of different architecture frameworks. Therefore, for the evaluation an attempt was made to consider a representative mix of architecture frameworks. The selection and grouping of the individual architecture frameworks are based on the opinion of the author and the classification as architecture framework in widely used literature (see section 3.4.5.2), however, does not aim to represent a complete listing of all globally available frameworks.

In preparation for the evaluation, the previously presented architecture frameworks will be summarized in tabular form in this section (Table 3) and in the annex (Table 9 to Table 16). The contents of the individual architecture frameworks are not presented in full extent. The focus is on specific aspects that are relevant with respect to the formulated requirements and the evaluation. The content presented includes the name, objective, and type of the architecture framework, as well as the application area, the focus on technical architecture content, utilization of reference architecture content in general, and transition support for deriving system architecture content from reference architecture content. In addition, a reference to the utilized sources is provided. The complete contents can be found in the sources indicated. As representative examples the enterprise architecture framework "Department of Defense Architecture Framework (DoDAF)" and, as a contrast, the system architecture framework "SPES_XT modeling framework" are shown in the following table. Those examples and related architecture framework kinds represent the range of the considered architecture framework mix from very abstract and broadly applicable to rather concrete and specifically applicable.

Table 3: Exemplary comparison of two selected architecture framework approaches

Name	Department of Defense Architecture Framework (DoDAF)	SPES_XT Modeling Framework
Objective	Alignment of all architectural descriptions created by various commands, services, and agencies to achieve compatibility	Provide framework for model-based systems engineering (MBSE) of embedded systems to professionals and practitioners concerned with the development of such systems in various domains
Type	Enterprise architecture framework	System architecture framework
Application area	Focus on military / aerospace domain and system of systems (SoS)	Focus on automation, automotive, and avionics – example: adaptable and flexible factory
Focus on technical architecture	Low	Medium
Utilization of RA or related content	None	Yes (not explicitly) – only mentioned as output
Transition support between RA und SyA	None	Yes (not explicitly) – abstract concept mentioned
Source	[28, 61, 132, 138, 140]	[33–35]

All architectural frameworks and the relevant literature considered are evaluated against the specified requirements. The specific content of the individual architecture frameworks is examined and evaluated in relation to the content of the other considered frameworks. The fulfillment of the requirements is divided into four levels. Two levels describe the extremes, i.e., either that a requirement is not fulfilled or that no statement can be made about the degree of fulfillment on the basis of the available literature/information, and that a requirement is completely fulfilled. In between, a distinction is made between barley and partially fulfilled requirements. The results of the evaluation are shown in Figure 26.

Architecture Framework * Enterprise architecture framework ** System architecture framework	Requirement 1	Requirement 2	Requirement 3	Requirement 4	Requirement 5
Department of Defense Architecture Framework (DODAF) *	◔	◔	◔	○	○
Federal Enterprise Architectural Framework (FEAF) *	◔	◔	○	○	○
Generalized Enterprise Reference Architecture and Methodologies (GERAM) *	◔	◔	○	○	○
Model -Based System Architecture Process (MBSAP) **	◔	◑	◑	○	○
Method Framework for Engineering System Architectures (MFESA) **	◑	◑	◑	◔	○
Ministry of Defense Architecture Framework (MODAF) *	◔	◔	◔	○	○
Reference Model of Open Distributed Processing (RM -ODP) *	◔	◔	○	○	○
SPES_XT Modeling Framework **	●	◑	◑	◔	◔
The Open Group Architecture Framework (TOGAF) *	◔	◔	◑	○	○
Zachman Framework *	◔	◔	○	○	○

○ Not fulfilled / not specified ◔ Barely fulfilled ◑ Partially fulfilled ● Completely fulfilled

	Consideration of
Requirement 1	development of architecture descriptions
Requirement 2	creation of technical architecture descriptions
Requirement 3	possible use of predefined architectural content
Requirement 4	transition from reference - to system architecture
Requirement 5	implementation of specific methodologies, structures, content related components

Figure 26: Comparative evaluation of 10 architecture framework approaches (concept adapted from [137])

It should be mentioned, although not explicitly queried in the form of a requirement, that the frameworks considered largely represent the specific components of an architecture framework described in section 3.4.4. With respect to the specified requirements, a relatively mixed degree of fulfillment can be determined for requirements 1-3, which are better fulfillment by the more specific system architecture frameworks in comparison to the more abstract enterprise architecture frameworks. This result was to be expected since the requirements describe a relatively specific issue on a similar abstraction level considered within the system architecture frameworks. For requirements 4 and 5, insufficient fulfillment was found for almost all

frameworks. The SPES_XT and MFESA frameworks form a small exception, which at least insufficiently consider the topic of reference architecture as specified within the requirements. In the overall consideration of all requirements, the integration of reference architecture concepts for the derivation of system architectures is hardly taken into account in connection with the architecture framework concept. Regarding the causes for this development different assumptions can be made, which are to be discussed briefly in the following. The first possibility why only brief attention to this topic within architecture frameworks is to be found, is lacking interest of relevant stakeholders and or no need for such a concept. In the eyes of the author, this conclusion could be drawn from a pure consideration of the architecture frameworks.

However, if a holistic view of the literature and the high relevance of topics such as reference architectures is considered, it can be assumed that there is a fundamental interest in this topic and that other reasons speak for the low consideration. From the author's point of view, it is more likely that a mixture of the low availability of suitable reference architectures (see section 3.3.3) in combination with the poor transition process description, presupposes the need for a high initial investment, which many companies have previously shied away from and therefore not sorely considered the use of architecture frameworks and reference architectures. Over the last few years, organizations started to consider such topics due to the rapidly changing competitive environment, their need for securing competitiveness over time, and the advantages which those concepts provide. This development can also be clearly seen in literature and in related trending topics such as reuse. In addition, a look at current research projects, such as CrESt, allows a similar conclusion to be drawn. Based on these considerations, the apparent lack of interest at first glance changes into a research gap that needs to be filled. In concrete terms, this means that the use of a reference architecture, the associated necessary procedure for creating a system architecture, and the implementation within existing or from scratch defined architecture framework should be investigated.

3.5 Conclusion on Transition between Reference and System Architecture Content Utilizing Architecture Frameworks

In summary, it can be stated for these chapters that the creation of the architecture as part of the basic engineering and partly detailed engineering has a lasting influence on the total life cycle costs and on the future orientation as well as the performance of a system to be developed. In order to keep the development and later use of such a system architecture as effective and efficient as possible, supporting concepts such

as reference architectures and architecture frameworks can be used. In this context, a reference architecture provides an architectural description for a group of systems or for a domain under consideration, which can then be (re)used for the definition of a system architecture description. A look at the literature reveals many abstract approaches with regard to the domain of production systems, as well as a few concrete approaches, such as the reference architecture presented from the CrESt research project. The template character of the reference architecture is often not differentiated clearly in literature and should be strictly separated from the architecture framework concept. In contrast to the reference architecture, the architecture framework does not describe concrete results, but rather the procedure for creating specific architecture descriptions. The architecture frameworks shall provide guidance and support for the affected stakeholders in an extent, that they can ideally focus to a large extant on topics that cannot be provided by the framework. Such topics would be the actual creative process of specifying a system. The accomplished evaluation of architecture frameworks resulted in the discovery of probable research potential regarding the integration of reference architectures and the actual methodical transition to the system architecture description. In the following chapters, this research gap will be examined and a proposal for a possible design of an architecture framework, which methodically considers this issue, will be introduced.

Excursus on the Topic of Reuse

In connection with the considered topic architecture framework many overlaps with completely independent research topics occur, which are either delimited from each other or excluded. A completely independent research topic, but often mentioned in the context of the architecture development and in particular in relation to reference architectures and the transition to system architecture, is the re-use of objects/architectural contents. This complex topic area should not and cannot be considered in detail in this work but shall also not be excluded in the following without a few relevant remarks. The topic of reuse occupies research and industry alike in the most diverse fields of application. Basically, besides specific considerations, some fundamentals for the reuse of artifacts can be read from the literatures, such as [5] and [137]. In the eyes of the author, the most relevant points with respect to this thesis are that reuse can only happen when the content available overlaps with the content needed to describe a system [137], reusable artifacts are created along the life cycle, and that they can be reused in different life cycle phases [5], and that the reusability is influenced by the degree of abstraction [5]. In addition to these points, there are countless other aspects that are considered in research contributions such as by [137].

4 Research Needs

In the previous chapters was shown that there are currently changes to be addressed in the context of systems engineering and in the subarea of the creation of architectural descriptions. Those occurring changes are expressed in different challenges for these disciplines. It could also be shown that there are specific research needs regarding the issue investigated in this thesis. The main is the use of reference architecture descriptions for the definition of specific system architecture descriptions through the application of a corresponding architecture framework. The considered research needs with respect to the topic of this thesis are described in more detail and narrowed down in the following in the form of research questions. Above all, the focus of the thesis lies on the specific challenge of "how" the transition from reference to system architecture can be implemented as part of architecting carried out by the relevant stakeholders of an organization. The goal is to provide a specific procedure within an architecture framework to support the concerned stakeholders, for the ultimate purpose of solving the challenges and capitalizing on the benefits.

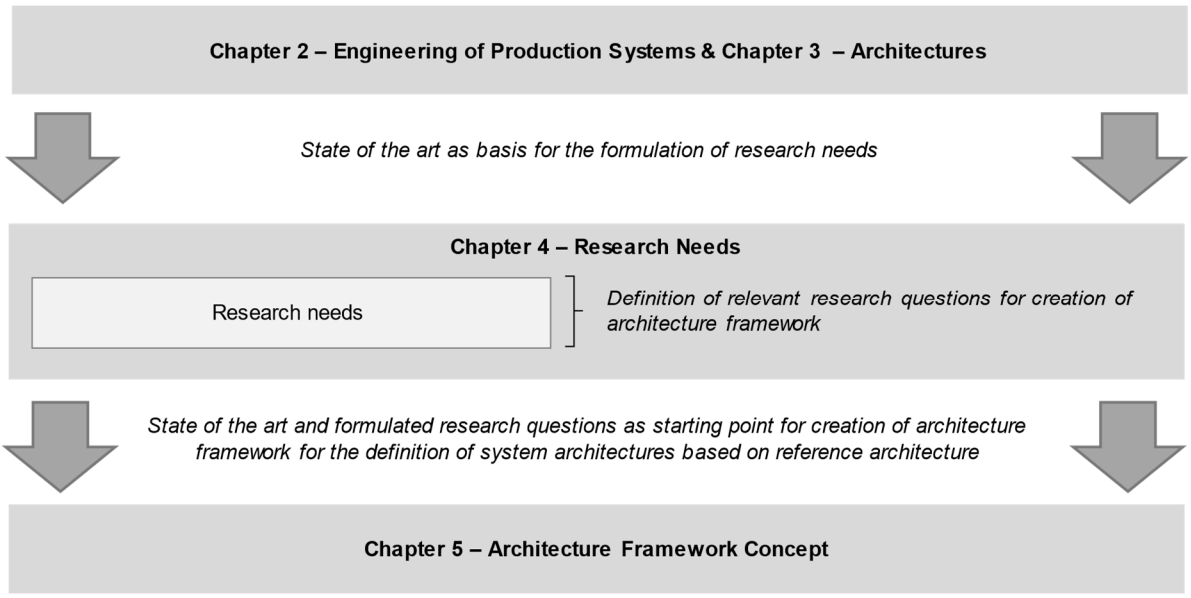


Figure 27: Overview of the contents of chapter 4

It should be mentioned at this point that the author is conscious that, as indicated in the previous chapters, the consideration of a holistic view of the organization, all life cycle phases, and related processes is more goal-prominent for the creation and application of an architecture framework. However, since many established architecture frameworks already exist and an integration of a specific framework into a widely use framework is possible under certain circumstances, the author decided to regard and work on a single relevant research issue, in order to provide a solution

for this research gap. The solution shall be used in the relevant community as discussion basis for the further consideration of the topic. The purpose is to initiate and advance the development in this area. If successful, the prototypical integration into a standardized architecture framework can be examined as a downstream task after the thesis.

In the following, the research questions of this thesis are presented and described. In addition to the main research question RQ1, other related and complementary subordinate research questions are defined (RQ2 and RQ3). The goal is to divide the problem space described by the research questions into smaller thematic blocks and to provide solutions for them. Chapter 4 thus formulates the research object of the thesis and as shown in Figure 27, represents the transition between the state of the art examined in the previous chapters and the architecture framework solution approach developed in the following chapter.

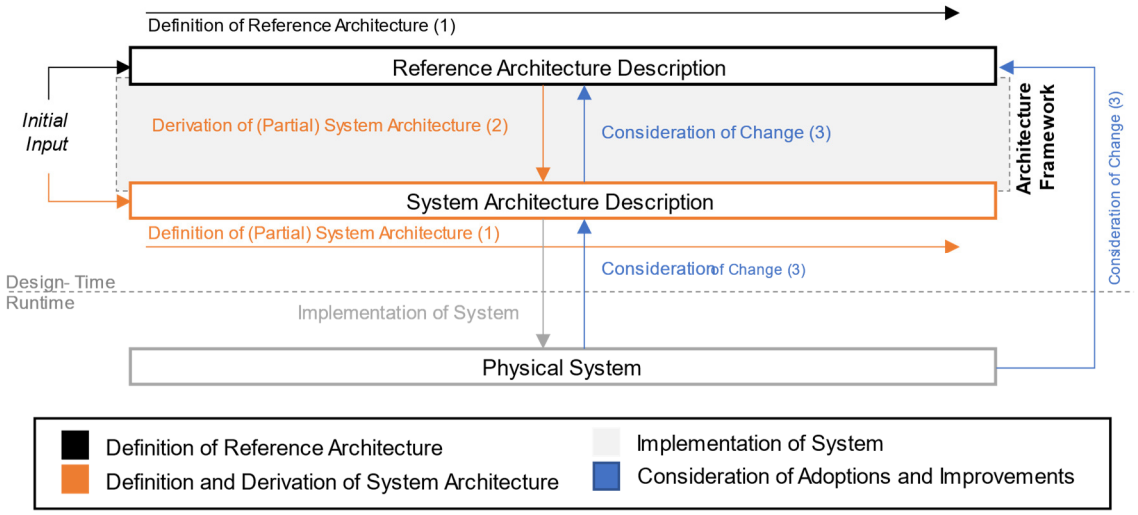


Figure 28: Model of transition from reference to system architecture (adapted from [146, 147] and [148] in [149])

Based on the state of the art and the research gap identified, a highly simplified model can be derived and used as a basis for formulating the specific research questions. The model is shown in Figure 28. The model describes the basic concept of defining system architectures based on reference architectures. Based on the model three areas of interest in relation to the research gap can be identified:

(1) In general, the method for defining architectures does not differ between reference architectures and system architecture, if they are considered and defined completely separated from each other. When setting up system or reference architectures only their scope as well as specific contents differ and therefore their applicability (compare with Figure 23 - single system vs. group

of systems). The applied methodology within the framework remains the same for both system and reference architectures (also as described in section 3.4.2).

(2) The validity of the made statement in (1) needs to be adapted, when system and reference architectures are considered in the same context. The methodology stays mostly the same but needs to be enhanced for considering mutual dependencies between reference architecture and system architecture. This is mainly the case during definition of system architectures based on reusable reference architecture elements as well as during adaption and improvement of architectures (see (3)).

(3) During the definition of a system architecture description or after the transition from design to runtime necessary changes to the system and therefore the architectural description can occur and need to be considered.

Based on these basic and highly simplified considerations and taking into account the research gap, the research questions are formulated.

The main research question RQ1 defines the general research challenge, which shall be tackled within this thesis. As shown in chapter 3 and especially in section 3.4.5.3, which covers the current state of the art regarding the architecting of systems, the transition between reference and system architectures is only partially covered in certain aspects within architecture frameworks. The relationships between system architecture and reference architecture as well as the role of the architecture framework are defined within literature. The idea of utilizing a reference architecture as a template for the definition of a system architecture is also defined on an abstract level. But the available knowledge does not provide a concrete procedure how such a transition could be carried out within an architecture framework. Therefore, this shortcoming is the main objective of this work. The purpose of RQ1 is to make a first push in the area of methodological guided transition between reference architecture and system architecture, thereby defining a first prototypically applicable concept, and enabling a basis for further research in this area.

Main Research Question - RQ1: *How to specify a system architecture description based on a predefined reference architecture utilizing an architecture framework?*

To comprehensively answer the main research question (RQ1) correlated sub-research questions are needed and defined in the following (RQ2 and RQ3)). The purpose of the research questions is to divide the problem space into smaller manageable parts, to structure them, to solve the challenges, and then to build on each

other to develop a solution for the main research question RQ1. The following research questions do cover the transition between reference and system architecture directly or necessary supplementary topics.

As shown in section 3.4.2, the creation of architectural content is based on the use of an architectural framework. This applies to both, a reference architecture and a system architecture. As documented later in the assumptions for the creation of the architectural framework (see section 5.2), it is assumed that the same architectural framework is utilized as the basis for both reference and system architecture and therefore both have the same structure and element types for modeling specific content. For this reason, research question RQ2 examines the issue of which structural, methodological, and content-related components must be included in such an architecture framework.

Research Question RQ2: *Which structural, methodological, and content-related aspects are required within a suitable and production domain focused architecture framework for the definition of architectural descriptions?*

As described in literature, a reference architecture can be used as a blueprint for deriving one or more system architectures if these can be assigned to the group of systems considered in the reference architecture (see section 3.3). After examining which basic components the architecture framework must have in order to create the corresponding architecture content (RQ2), RQ3 examines how a transition between reference architecture and system architecture can be methodically designed and integrated into the architecture framework. The method must take into account inputs from both the reference architecture and the system architecture. Consequently, the following research question arises:

Research Question RQ3: *How could a possible methodological consideration of the transition between reference and system architecture look like within the architecture framework? And how can the methodological components be linked to the elements of research question 2 so that a holistic framework approach will result in the end?*

In the following chapters, the results regarding the research questions are elaborated and answered in summary in the concluding chapter.

5 Architecture Framework Concept

In this chapter, potential solutions for the research questions RQ1-RQ3 posed previously are addressed. The goal is to define a suitable architecture framework concept that can be applied reproducibly for the creation of system architecture descriptions based on reference architecture content. The purpose for the creation of such a concept is to investigate the research gap outlined in the previous chapters in detail, to address the related challenges in the field of engineering as well as architecting, and ideally to solve them or reduce their influence on the design process of a system. In order to do so, several topics as shown in Figure 29 and described in the following are considered. Based on the insides of this chapter the architecture framework concept is prototypically implemented within a modeling tool in the next chapter.

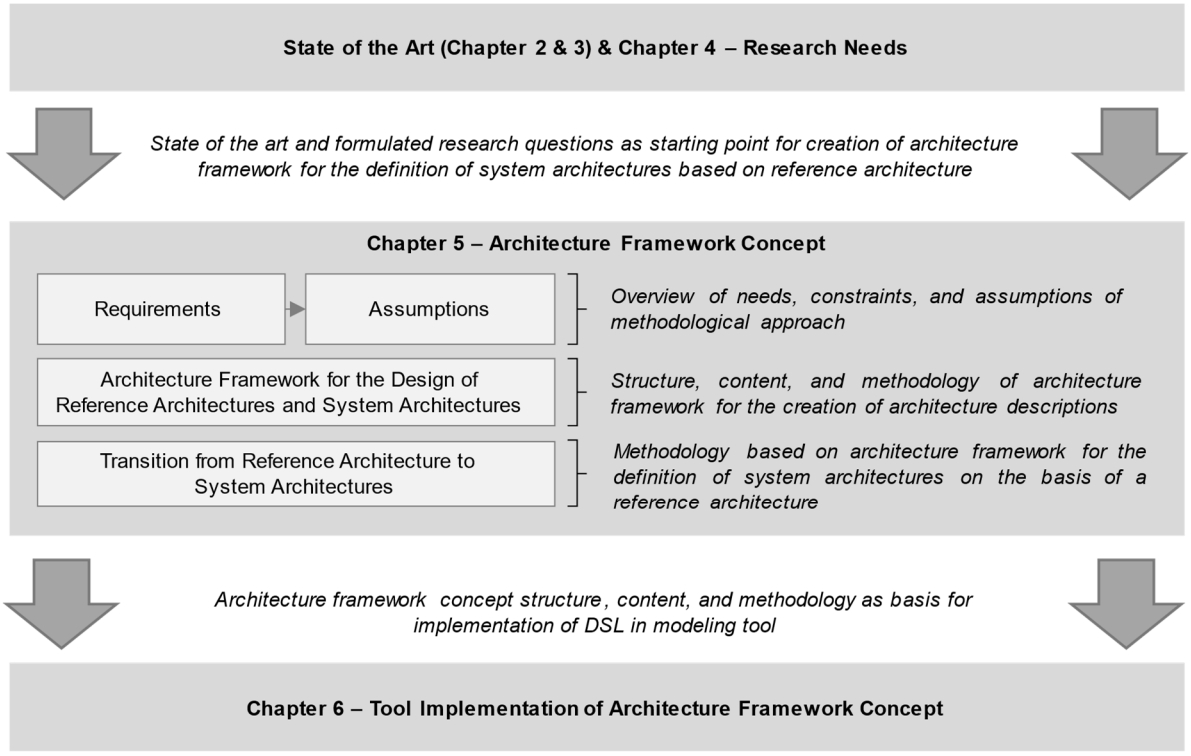


Figure 29: Overview of the contents of chapter 5

First, basic requirements for the development of the architecture framework concept and related assumptions are defined in sections 5.1 and 5.2. Both restrict the problem as well as the solution space and provide information about contents which are not explicitly considered but are assumed as given. Additionally, as a basis for the creation of the architecture framework the concepts of abstraction and granularity are defined in section 0. This is followed by section 5.4 concerned with topics regarding the basic

structural components and the methodological approach for the creation of architecture descriptions within the core architecture framework concept, which mostly tackle concerns of RQ2. Furthermore, in section 5.5, the core architecture framework concept is supplemented with the architecture framework transition method, which considered the utilization and integration of reference architecture contents into the process of the system architecture description creation. The transition method is mainly reflecting concerns of RQ3.

In summary, the main contents of the architecture framework concept derived from the research questions and the intended application of the framework are shown in Figure 30. The figure shows that the architecture framework concept units the core architecture framework concept and the architecture framework transition method. The core architecture framework concept can be applied for the individual creation of architecture descriptions for single systems or a group of systems and considers the general structure, relevant element types, as well as supplementing methodology for the creation of architecture descriptions. The architecture framework transition method explicitly considered the transfer between an already defined reference architecture description to a shall be defined system architecture description. Together, within the architecture framework concept, they can be applied for the definition of a system architecture description utilizing a reference architecture description.

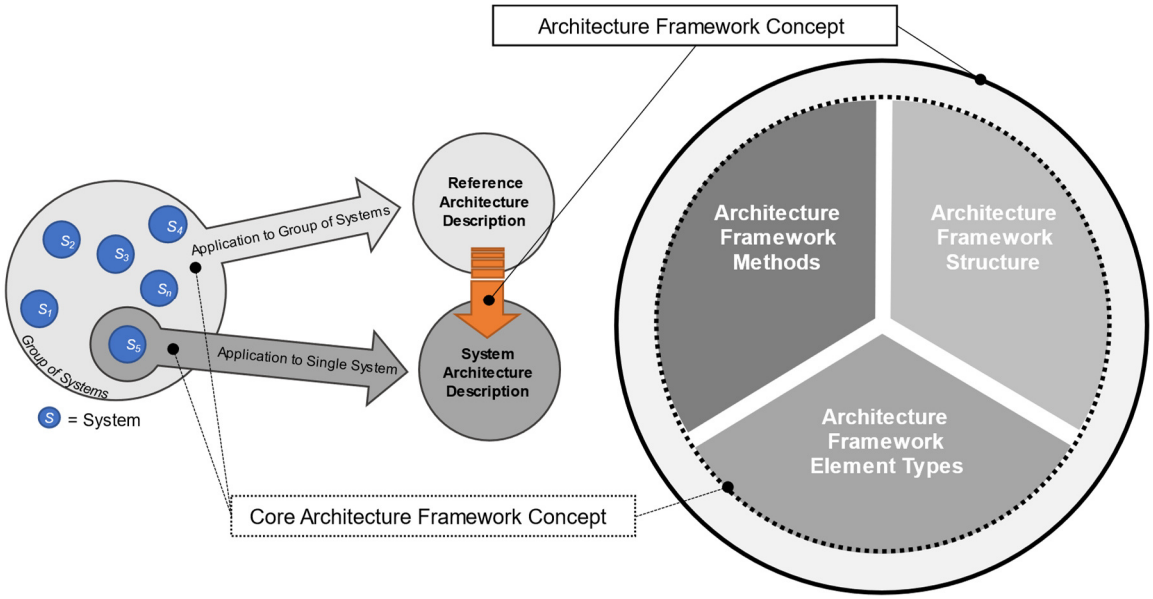


Figure 30: Abstract representation of the planned architecture framework concept

5.1 Architecture Framework Concept Core Requirements

Taking into account the relevant ISO guidelines and standards, such as VDI 3695, the contents and results of the CrESt research project, the presented state of the art, as well as the discussion with experts of the scientific field under consideration, the following core requirements for the architecture framework result from the author's point of view. The specification of the requirements is based on the current state of the art and the purpose and field of application of the architecture framework. The requirements are documented in order to ensure that all subjects of the research questions are considered within the concept. Furthermore, the defined requirements are used to evaluate the prototypical application of the architecture framework concept and its results to provide a meaningful conclusion. It should also be noted that the requirements are not to be understood like traditional requirements, such as defined by [90] or [150], and are to be seen more like indications for the development of the core contents of the architecture framework concept in this thesis. In addition should be said that, only the core considerations are taken into account and that the set of specified requirements is not complete. Further potential requirements, for example, regarding the content of frameworks as documented in the state of the art, are not explicitly mentioned. The requirements are clustered by the realms application/utilization, procedure, structure, and content in Table 4.

Table 4: Requirements for architecture framework concept development

Requirements: The architecture framework shall…	
Application /Utilization	• be applicable within the domain of discrete manufacturing systems. • consider architects concerned with the development of system architectures based on reference architectures as the main stakeholder(s) for its' utilization.
Procedure	• provide a procedure for the general definition of architecture content (system and reference). • provide a procedure for the transition between viewpoints and views. • provide a procedure for the definition of system architectures based on reference architectures. • provide a procedure for consideration of changes and potential adaptions to architecture content.

Requirements: The architecture framework shall…	
Structure	• consider the concept of problem and solution space. • cluster behavioral and structural aspects considered within the framework separately. • be utilizing the viewpoint/view concept for the definition of relevant architecture content (highlighted in the state of the art). • structure content of different viewpoints/views by their degree of detail using a granularity layer concept.
Content	• provide predefined model kinds and element types utilized within typical architecture related viewpoints applied in the basic and detailed engineering of a system, which are concerned with the domain of one or several systems, requirement(s) to the system(s), function(s) provided by the system(s), logical component(s) realizing the function(s), and the technical solution(s) realizing the specified requirement(s). • provide predefined model kinds and element types utilized within related views of the specified viewpoints, which are concerned with topics like the domain description and the related context of a system, specification of needs, stakeholders and potential use cases, consideration of potential products and their production processes, as well as the specification of resulting requirements, functions, logical components, and technical solutions. • provide a concept for consistent tracing of relationships between architectural elements across the whole architecture model.

With regard to the defined requirements, it should be noted that the classification of the content is not free of overlaps and that the remaining specified requirements should always be taken into account when considering a single requirement.

5.2 Architecture Framework Concept Assumptions

Before and during the creation of the architecture framework concept the assumptions listed below have been made by the author based on the contents and results of the CrESt research project, relevant ISO guidelines and standards, such as VDI 3695, the discussion with experts of the scientific field under consideration and the presented state of the art. These were made to limit the solution space to essential aspects and to show the influences under which it can be assumed that the created concept works. It shall be noted that the assumptions do not automatically eliminate the possibility that the concept can be used successfully outside of the defined solution space. In

order to be able to make a clear statement about the applicability and potential adaption to the framework for it to be used in other domains, further investigations and, if necessary, a prototypical application are needed. Such verifications are not part of this work, but potential points for further research. The assumptions made are listed in Table 5. The latter are clustered with respect to the topic architecture framework, system architecture, reference architecture, as well as to organizational and stakeholder related aspects. Even though different clusters are defined, they are all to be seen in relation to the architecture framework concept.

Table 5: Made assumptions with respect to architecture framework concept

	Assumptions
Architecture Framework	• Definition of an architecture framework for the creation of system architecture descriptions based on reference architecture descriptions is necessary because the current state of the art does not cover the transition from reference to system architectures in a sufficiently manner (see chapter 3.4.5). However, some general inputs regarding structure and content can be utilized from existing architecture frameworks. As a basis within this thesis the SPES_XT modeling framework is used. The main reasons for this are that the focus is already on the domain of the adaptable and flexible factory and the corresponding basic structures already exist. Furthermore, a reference architecture that can be used for the evaluation has already been defined using the framework (see section 3.3.3). • The designed architecture framework concept mainly focuses on the design time of a production system, especially on the basic engineering, partially on the detailed engineering (architecture related topics only), and the engineering related topics along its life cycle. • The governance for quality assurance and maintenance of the architecture framework concept is in place and is therefore not considered further (see for example [28]). • The abstraction level is set before applying the architecture framework concept for the creation of architecture description. Focus is only on structure and granularity layer concept. • For the transition and better comparability of contents, the existing refence architecture structure and used element types are the same as introduced by the architecture framework concept for the specific to be defined system architecture.

Assumptions	
Architecture Framework	• Requirements are considered to be view overarching and it is assumed that they are constantly monitored and considered by the stakeholder creating an architecture description. Therefore, they are not considered as direct input for every view within the method. The directly neighboring views of the requirement viewpoint views shall consider the contents as direct input.
System Architecture	• Only new, nonexistent systems are considered as systems to be designed. The migration of existing systems to an upgraded version of a system is not considered, as this topic has already been covered in literature researching “migration” (see, among others, [151]).
Reference Architecture	• The reference architecture utilized for later prototypical application is usable for the domain of production system as well as for discrete manufacturing and shares common content with the to be defined system. • The reference architecture, which should be used as a basis to derive a specific system architecture description, is created with the same architecture framework as applied for the derivation of the system architecture. A different architecture framework would create different contents within reference and system architecture, which might not be compatible or at least could require an additional matching and sorting procedure. Within this thesis this additional step is not carried out to simplify the introduced procedure. • Systems engineering and its concepts, such as architecture frameworks, are accepted, considered useful, and beneficial by the whole organization and are supported by all relevant stakeholders and the management as well. Therefore, necessary organization transformation and systems engineering implementation strategies are not considered within this thesis.

Assumptions	
Organizational and Stakeholder Related Aspects	• The utilization of a reference architecture and the associated architecture framework concept for the design of system architectures is economically and from a company point of view reasonable. When considering the main benefits of architecture frameworks this assumption is reasonable if not only one system architecture shall be created, but several with a focus on the same domain. Otherwise, the application of an architecture framework/reference architecture for the definition of system architectures would not be target-oriented as well as more cost and time consuming as a creation of a system architecture from scratch. • The designed architecture framework concept is used by a well-educated and SE experienced audience. Despite all efforts to make the method as simple and understandable as possible, a certain amount of implicit knowledge and creativity is required for a successful application. Users with little experience usually do not have this knowledge, e.g., to decide how a system can logically be composed in a way that it can be easily transferred into a technical solution. Therefore, with respect to the thesis the concept mainly targets experienced engineers and system architects. In the long term, the goal shall be to make the concept accessible for all users to be able to achieve good results even with a basic level of knowledge. • The cooperation between different stakeholders and between or across different disciplines, which is necessary for the design of a complex system, works smoothly. An architecture framework can also contribute to the optimization of communication, documentation of knowledge, and the traceability of decisions made. However, the thesis does not focus on the optimization of structural challenges and therefore, despite its importance, those are not considered. • The applying stakeholder basically agrees with the content, structure and arrangement of the architecture framework concept, and the utilized reference architecture. If this is not the case and large parts of the reference architecture content or of the architecture framework concept would be, without necessity, adapted and additional effort would be spent, the use of reference architecture and architecture framework concept loses its impact. In such a case the system architecture could also be defined directly from scratch without spending additional effort on adjustments.

5.3 Abstraction and Granularity in Context of Architecture Framework Concept

Before the (core) architecture framework concept is introduced, the abstraction level and granularity layer (see definition, section 2.1.2) should be introduced and differentiated from each other. This consideration is essential for the overall understanding of the utilization of reference architecture content within the architecture framework concept for the definition of system architectures. As defined, during the procedure of creating a reference architecture a specific domain or group of systems is always considered. To utilize the reference architecture within the architecture framework concept, the specified content of the reference architecture must be usable to describe the system of interest considered within the system architecture to be created. In this context, the abstraction level plays an important role for the validity assessment, whether a reference architecture considers a group of systems to which the system of interest can be allocated and therefore content can be reused. These considerations are particularly important for the utilization of a reference architecture in an architecture framework concept. For this reason, the term abstraction is examined in more detail below and distinguished from the term granularity, which is often erroneously used synonymously. The described relationships are simplified shown in Figure 31.

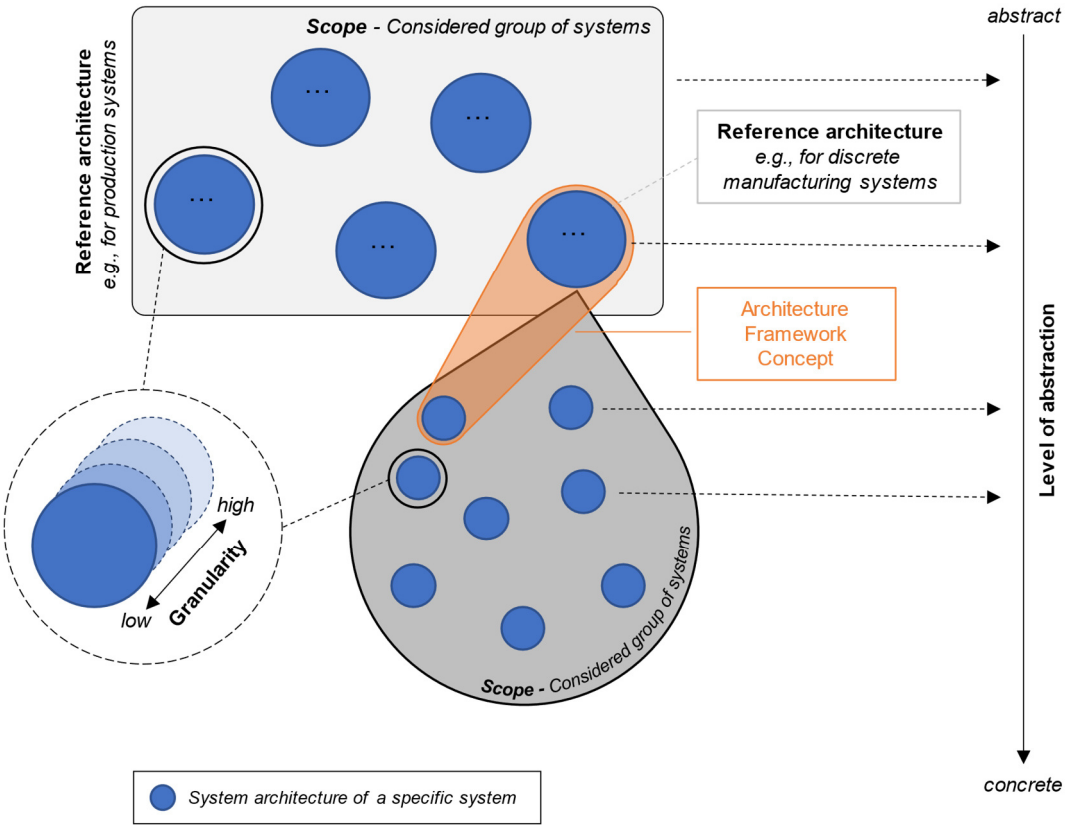


Figure 31: Abstraction and granularity in context of architecture framework concept (adapted from [147])

According to the Cambridge Dictionary the word “abstract" can be defined in the conventional sense as "[...] referring to something which exists as an idea and which is not physically real" [152]. The abstraction levels, which range from abstract to concrete, should enable a classification of considered systems. Based on the definition, a system at a high level of abstraction (less concrete) can be regarded as a concept that is close to reality. A system at a low abstraction level is a concrete system that can be technically implemented. It is important to note that the range of abstraction depends strongly on the application scenario as well as on the involved stakeholder(s) and related goal(s). Through the abstraction levels, systems can be set in a relationship to each other. From now on, abstraction levels are considered in this thesis as follows.

Level of Abstraction

The abstraction level relates systems of a particular domain/environment to each other and provides information about their level of detail in relation to other systems within the domain/environment. That is, to place them in a span between abstract and concrete system classification. Following the ISO 9000 standard [153], systems at a low level of abstraction inherit all the characteristics of higher-level concepts and contain information that distinguishes them from systems at the same level.

Each specific representation of a system at a certain abstraction level has a corresponding architecture. When considering a range of abstraction levels, the same system, previously considered as a system architecture, can then also represent a reference architecture if that system represents a group of systems at lower abstraction levels. For example, if the representation of a discrete manufacturing system shown in Figure 31 is considered as an isolated single system at a specific abstraction level, it can be referred to as a system architecture. If the same system is considered as a representation of a group of systems, assigned to lower abstraction levels, a reference architecture is represented. Furthermore, regardless of the level of abstraction, any system can always be detailed using the granularity as defined in section 2.1.2. It should be noted that the terms abstraction and granularity are partly used synonymously in the literature. In the context of this thesis, the two terms are considered separated from each other. Therefore, each individual system can be assigned to a certain abstraction level and has a certain granularity. This separation takes its orientation from design space concepts like the one described in section 2.3.3. It can be stated that the abstraction level describes a system from an external

perspective and the granularity describes the interior of a system as well as the level of detail that naturally results from its internal structure. When looking at Figure 31, it can be stated as an example for abstraction that a production system is more abstract than a concretization in the form of a discrete manufacturing system and that the application example of cylinder head manufacturing used within this thesis is more concrete than the discrete manufacturing system (as shown in Figure 5 and section 1.4). The granularity, on the other hand, describes the content-related detailing of a system of interest. For example, the cylinder head manufacturing can be further specified into a manufacturing system, a transport system and a control system on a more detailed granularity layer but at the same abstraction level. The total scope of the abstraction and the associated abstraction levels depends, as described, on the domain or system group of interest and the resulting reference architecture. Depending on the focus, these levels are comprehensively defined in literature. With regard to production systems, different sources in literature, such as [5] and [137], define different hierarchy levels that reflect potential abstraction levels with respect to the domain. For example, the levels described by [5] include, at the highest level, the production network, followed by the factory, the production line, the production line segment, the work unit, the work station, the function group, components and finally the construction elements. Depending on the level of abstraction and the context boundary, conclusions can be drawn about the main task of the system. Those tasks turn out differently at different levels.

Production System Application Example

For the exemplary consideration of abstraction and granularity a simplified thought model is created. Within the model, it is assumed that a cylinder head manufacturing system, as a reference architecture, consists of 2 + 4 additional architectures. Namely, of systems which consider the manufacturing of a cylinder head with 4 cylinders and a cylinder head with 8 cylinders and each cylinder head (4/8 cylinders) will be produced with three or five valves per cylinder. Using this very simplified example, the relationship between abstraction and granularity is shown in Figure 32. Based on the thought model and the considerations made within the figure the following statements emerge:

a) The degree of abstraction describes the level of concretization for the definition of a reference or system architecture under consideration.

b) Starting from the abstraction level of the reference system, increasing granularity leads to subsystems with a lower abstraction, which means, when starting from a very high abstraction (cylinder head manufacturing system), due to the increasing granularity, a lower level of abstraction can be achieved (e.g., cylinder head manufacturing - 4 cylinder/8 cylinder).

c) If one system is described on continuously detailed granularity layers, the derived subsystems become more and more concrete, i.e., granularity layers ➔ ∞ provides degree of abstraction ➔ 0, i.e., 100 % concrete description, and the number of systems will also ➔ ∞.

Those statements can be interpreted with respect to abstraction and granularity and lead to the conclusion, that an identical system (e.g., (2b) Cylinder head manufacturing 8 Cylinders) can be considered either as reference architecture (RA) with lower abstraction level compared to another RA (e.g., (1) RA of "cylinder head manufacturing") or as sub-system on a higher granularity layer of a RA (e.g., (2b) is on granularity layer 2 of the (1) RA "cylinder head manufacturing").

Depending on the point of view, it can also be deduced from the simple example that a reference architecture can consist of reference and system architectures with a lower degree of abstraction. The challenge is that for real system architectures, these relationships must be considered in a n-dimensional space.

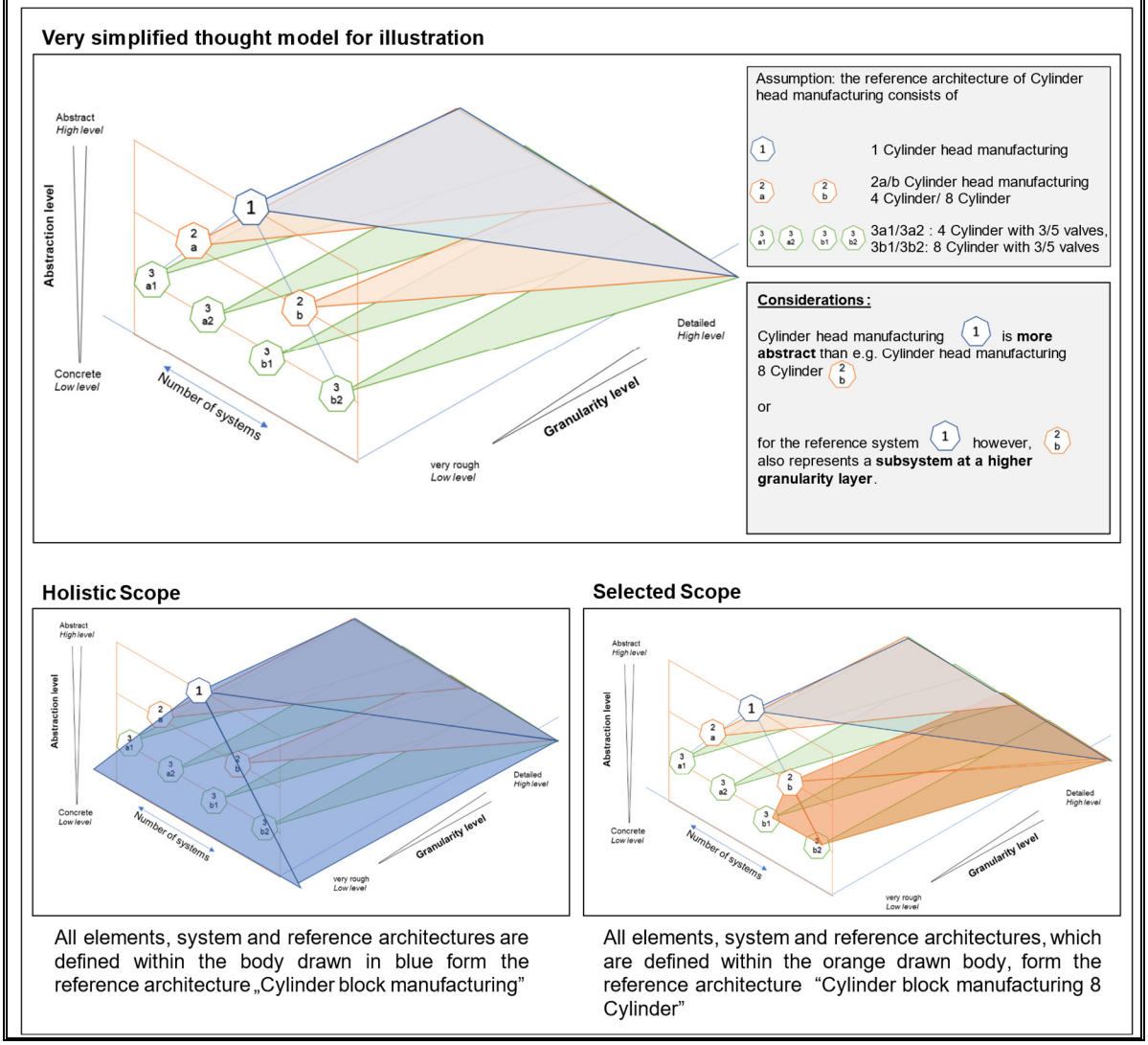


Figure 32: Thought model for explanation of relationship between abstraction and granularity

Drawn Conclusions regarding the Architecture Framework Concept

The architecture framework concept is used to create specific system architecture descriptions for systems from the group of systems considered within the reference architecture, as shown in Figure 31. The specified reference architecture content, which is detailed and correlated using the granularity layers, is then used to derive the system architecture description. Based on the concept shown in Figure 31 as well as in Figure 32 and taking into account the intention of using an architecture framework in combination with a reference architecture three main consideration can be derived:

(1) The transition of system content is only to be brought into a context as long as the systems are part of the same domain, stand in a relationship with each other, and share common content. The consideration is partly drawn from [137] and the shown connections between requirement fulfillment and reuse. This means that a reference architecture within the architecture framework concept can only be applied to systems that have the same or a lower degree of abstraction.

(2) A big difference between the abstraction levels of the systems of interest and the available reference architecture description results in abstract reusable content, from system abstraction point of view, which can be most likely reused in main parts but will also result in additional efforts for required detailing and supplementation of available content. Depending on the application scenario, this tends to have a negative impact on the usability of the reference architecture [89].

(3) With increasing concreteness, taking into account the absolute abstraction layer observation space, the amount of detail in the granularity layers increase.

Based on these considerations, a general relationship can be derived between abstraction level differences, reusability, and the need for adaptation and supplementation of contents from scratch when using a reference architecture for the definition of specific system architectures. These relationships will not be explored further in this thesis, as simplified assumptions were made for the architecture framework concept. Nevertheless, the connection between the relationships should be investigated in the future, as it may be of great relevance, for example, in the context of assessing the quality of applicability of a reference architecture.

5.4 Core Structure of Architecture Framework Concept

In this section the general methodology, structure, and content of the core architecture framework concept for the definition of architecture descriptions is introduced. Dependent on the scope, the architecture framework can be utilized for the definition of reference architectures and systems architectures from scratch. The architecture framework concept forms the basis for the transition between available reference architectures and to be defined system architectures. The transition method for the definition of system architectures based on a reference architecture will be introduced in section 5.5.

5.4.1 Scope of Architecture Framework Concept

As already addressed in section 2.2 in the context of the life cycle description of a system, this thesis and the architecture framework concept focus on the basic engineering and partially with respect to architectural considerations on the detailed engineering. The structure and concept of the core architecture framework concept is based, among other things, on the state of the art as well as the specified requirements and the assumptions made. In terms of structuring the contents of the core architecture framework concept, the framework takes its orientation from the VDI 2206 and utilizes macro and micro cycles. The macro cycle describes the overall considered procedure and is based on the classic steps of basic/detailed engineering from the consideration of domain and requirements to the creation of a first technical solution architecture, as for example indicated in [154] based on [155]. In terms of the core architecture framework concept those steps are captured within the resulting structure, the predefined model kinds, and element types of the concept (described in section 5.4.2). The micro cycles describe a set of methods recurringly applied during the processing of the macro cycle. With respect to the core architecture framework concept the associated micro cycles and rules presented in section 5.4.3 describe how and when content is defined and adapted. In combination, the micro and macro cycle shown in Figure 33 represent the main components of the core architecture framework concept described in Figure 30. The core structure of the architecture framework concept is later extended by the architecture framework transition method as described in section 5.5.

In order to point out the scope of this thesis and the developed framework the architecture framework concept is exemplarily allocated to a well-known system engineering process for developing a system. Figure 33 shows the scope of the thesis with respect to the V-model of the VDI 2206. In comparison to the full V-model, in which many stakeholders fulfill different roles and tasks in relation to the development

of a system, the focus of the architecture framework concept is mainly on system architects or engineers who are involved in the creation of a system architecture as part of basic and detailed engineering.

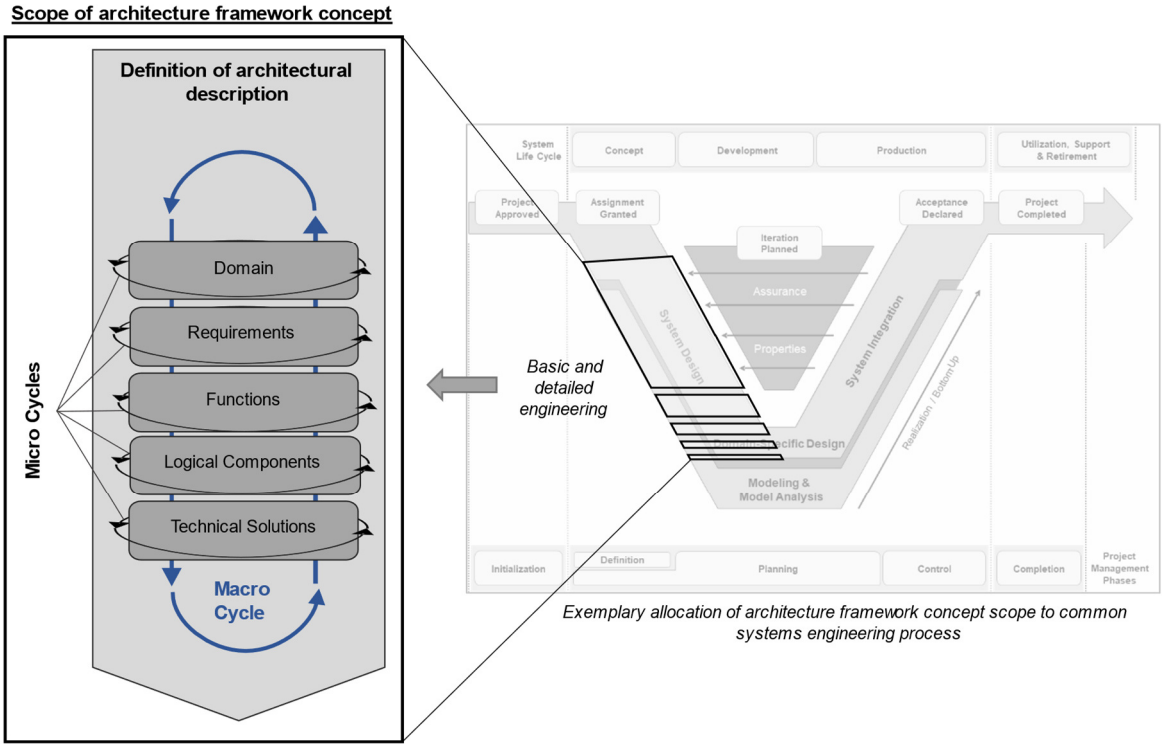


Figure 33: Scope of architecture framework concept

5.4.2 Macro Cycle - Structure and Content of the Core Architecture Framework Concept

The contents of the defined macro cycle considered within the core architecture framework concept are, as mentioned above, closely related to the basic engineering and the architectural considerations of the detailed engineering. These contents of the macro cycle are mostly derived from identified SPES_XT modeling framework, as well as from relevant literature and standards, and will be utilized for the specification of the core architecture framework concept in the form of structure, predefined element types, and methodologies. The macro cycle representing the procedure for creating an architecture description is represented in Figure 34. Within the figure, the macro cycle is depicted as an iterative procedure which, starting from an input in the form of a problem, creates a domain and requirements description, followed by deriving a functional, logical, and technical architecture. The derivations of the individual architectures always stand under the premise to fulfill the defined requirements. The requirements are considered by the functions provided by the system. Those functions

are realized by different logical components as well as finally technical solutions. As shown in Figure 33, the technical architecture description represents the result of the macro cycle, which forms the basis for further detailed design and implementation.

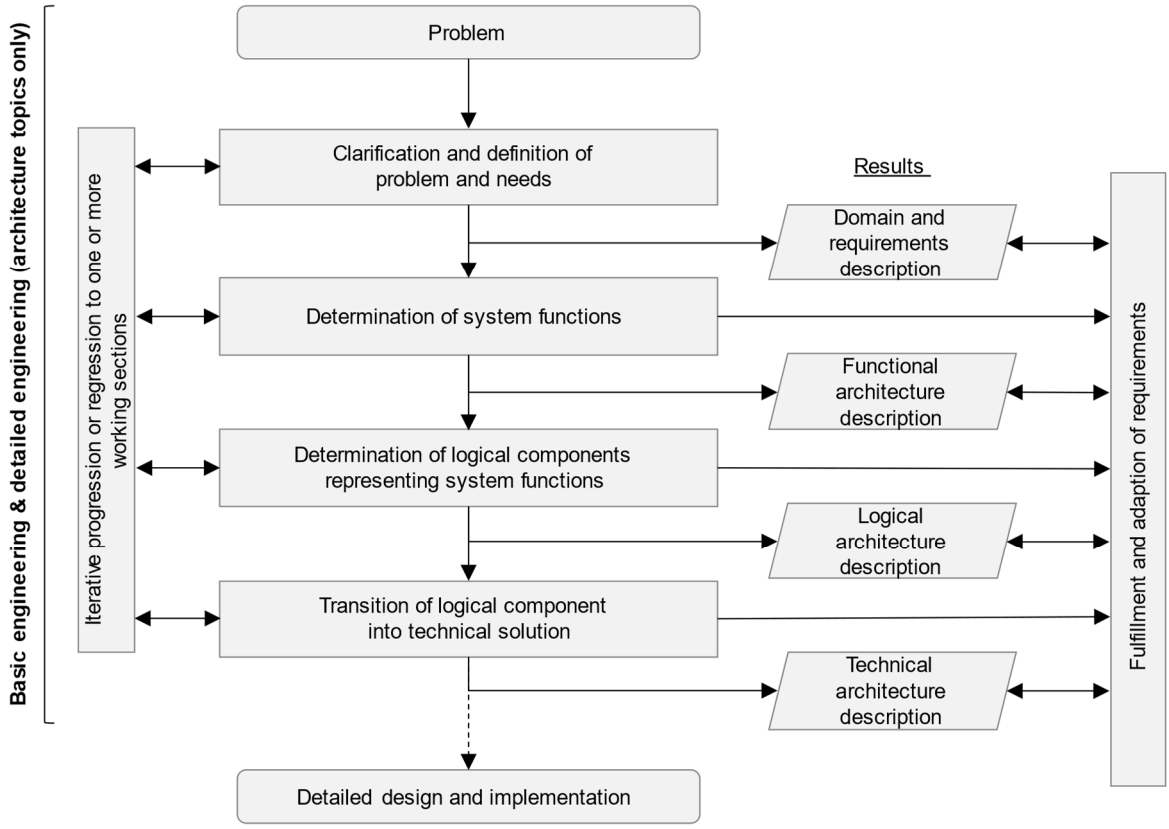


Figure 34: Representation of macro cycle (adapted from [154] based on [155])

Based on the macro cycle and associated insides, in the following, the core architecture framework concept will be specified. First, the core architecture framework concept is structurally divided into a problem space and a solution space. Secondly, the core architecture framework concept is supplemented by interconnected viewpoints, which are assigned to the problem and solution space based on their content and represent the considerations made within Figure 34. Thirdly, the defined viewpoints are supplemented by related levels of detail. This allows the content within the viewpoints as well as in the problem and solution space to be viewed at different layers of granularity. Fourthly, the different views within a viewpoint and on a certain layer of granularity are introduced and assigned. Finally, the predefined element types and model kinds are described.

5.4.2.1 Problem and Solution Space

Within this section, the concept of problem and solution space will be defined to be able to cluster the viewpoints of the core architecture framework concept with respect to those spaces. This division makes it noticeably clear to the user, if the overall problem/goal of the system or a solution to resolve the previously defined problem is considered and defined within a viewpoint. This distinction is crucial for a comprehensive and successful engineering of a system. Appropriate solutions can only be defined when the problem has been fully identified and defined in its entirety. The clear boundaries allow to monitor if the defined solutions address the main problem and all defined sub-problems. Only then a system can be implemented and operated successfully with minimal adjustments.

In general, a design space always describes a complex all-encompassing space of all possible system design solutions with respect to the problem/goal [156–158]. However, only the more precise solution space and the dependent problem space are considered in the following. To visualize the two concepts, in Figure 35, the Twin-Peaks model as a representation of the importance of the relation between creation of requirements and architecture design, initially introduced by [159] (see [160]) and refined by [160] and [161], is shown. This model was supplemented within [161] by a problem and a solution space, which clearly shows the borders but also the connection between the two spaces. The solution space represents a part of the design space (best fit) but does not comprise it completely [158]. The problem space defines, among others, relevant requirements as well as stakeholder needs and goals regarding the system of interest [162, 163]. It considers not only the future system but also the related stakeholders and surrounding environment of the system. The solution space represents all designs and results with respect to the defined content of the problem space. The progressive specification as an iterative process between problem and design space extends over the different levels of granularity. Depending on the consideration, the created architecture description, in the problem space, is rather realization independent and in the solution space rather realization dependent.

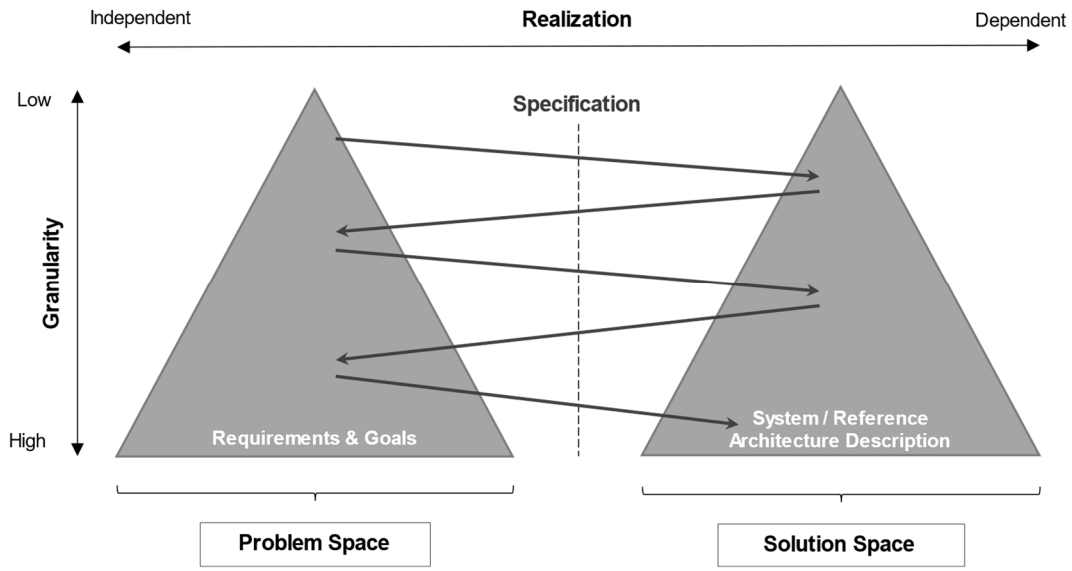


Figure 35: Characterization of problem and solution space - adapted from [159–161]

5.4.2.2 Core Architecture Framework Concept Viewpoints

As already described before, the viewpoints that are considered in the context of the core architecture framework concept result from the preceding considerations and introduced contents. At this point, reference should be made to the input of the SPES_XT modeling framework [33, 34] used as a reviewed and in the community accepted basis for creating the structure of the architecture framework concept. The SPES_XT modeling framework has already been considered during the assessment of architecture frameworks in section 3.4.5 and was classified as a usable starting point in the assumptions (see section 5.2). Based on the assessment and selection, the core architecture framework concept adopts the basic structuring into requirement, functional, logical and technical viewpoint, which is common for basic and detailed (systems) engineering as shown in Figure 34, as well as the basic principle of granularity from the SPES_XT modeling framework. All other structural, methodological, and content-related concepts within or outside of the viewpoints were created separately in the context of this thesis or in the context of the author's work on the CrESt project.

The core architecture framework concept introduces a total of five different viewpoints. In addition to the four viewpoints mentioned above, the problem space consideration is enhanced by including the specific domain as shown in the macro cycle. The domain and requirement viewpoint are assigned to the problem space. The remaining three viewpoints, functional, logical, and technical viewpoint, are part of the solution space. The fixed order results from the content sensitive relationships between the viewpoints. This means that necessary functions, logical components,

and derived technical solutions can only be defined meaningful and complete once the domain of the system and the requirements for the solution have been specified. The transition between and within a viewpoint is described in 5.4.3. The allocation of viewpoints within the architecture framework concept to the problem and solution space is depicted in Figure 36. The description of the viewpoints and their connections are described below.

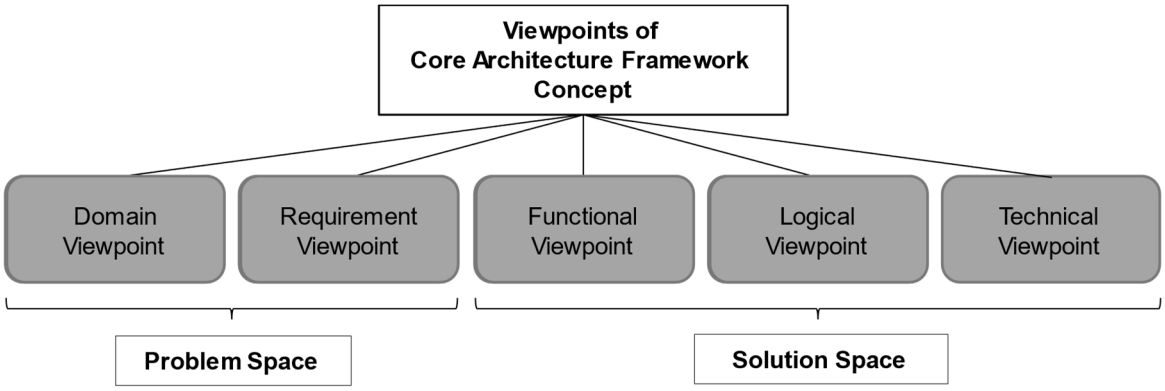


Figure 36: Definition of viewpoints of core architecture framework concept and allocation to problem/solution space

Domain Viewpoint

The first viewpoint of the architecture framework concept and part of the problem space is the domain viewpoint. The goal of this viewpoint is to provide a first overview of the system or group of systems of interest and to define its context, its environment, and its stakeholders within the considered domain. A domain can be defined as follows.

Domain

A domain can be defined as a "[...] sphere of knowledge, influence or activity [⋯]" [164]. With respect to [164] the subject area to which a stakeholder applies systems engineering procedures and methods is the domain of the system of interest.

The domain viewpoint documents all known and relevant interfaces between the above listed elements of the domain. The purpose of the domain viewpoint is to describe the system and all related elements as precise and complete as possible so that comprehensive requirements can be derived. In the viewpoint all unilateral or mutual relationships between elements should be shown, so that a conclusion can be

drawn which elements influence each other and which do not. This is important to ensure that any system-relevant effects are considered properly, and none are forgotten during further architecting steps.

Requirement Viewpoint

The second viewpoint of the core architecture framework concept and part of the problem space is the requirement viewpoint. which is based on the domain viewpoint. The ultimate goal is to define what the system does, how it performs these tasks, and what constraints may need to be considered in the form of requirements. A requirement can be defined as follows.

Requirement

A requirement can be defined as "[...] a statement concerning a property or the performance of a product, a process or the people involved in the process" [165]. In addition, a requirement describes a condition or capability to be provided by a system and needed by a stakeholder to solve a problem [41].

In order to be able to make these definitions, the goals and needs of the relevant stakeholders, possible use cases, and the product to be manufactured are examined and defined [15]. These preliminary considerations serve to describe the requirements of the system of interest. The system is specified to provide a specific performance/service with the purpose of meeting the defined needs and objectives of the customer, operator, or other stakeholders [30]. The requirement viewpoint represents the interface to the solution space and the transition to the functional viewpoint.

Functional Viewpoint

The functional viewpoint represents the first viewpoint within the solution space. This is where the transition from the problems and requirements defined in the previous viewpoints (domain/requirement) to the solution-oriented functions takes place. The term function is specified below.

Function

A function is an action or task provided and executed by a system with the aim of fulfilling the goal and the defined purpose for which the system has been created [82, 166].

The goal of the functional viewpoint is to define how a system of interest and its provided functions can fulfill the requirements and the needs of the stakeholders specified within the problem space. These solutions are defined in the form of implementation-neutral functional descriptions. The functional architecture can then be tested as the first output of the solution space against the specified needs of the problem space and adapted if necessary. Depending on the literature, logical and/or technical solutions are derived for the specified functions from which the system of interest is composed. It seems that the intermediate step of the logical architecture is becoming more and more relevant due to the technology neutrality in connection with reference architectures [28] and the reuse of elements.

Logical Viewpoint

The logical viewpoint is the subsequent viewpoint in the solution space and describes the realization of specified functions within logical components. The term logical component is defined as follows.

Logical Component

A logical component represents the realization of one or more required system functions on a solution-oriented, technically implementation-free level.

The functions are then converted into technically implementation-neutral solution elements that represent the system of interest and shall fulfill all specified requirements. These implementation-neutral logical components can then be reused in whole or in parts within a reference architecture, or they can be converted into specific technical solutions for the creation of a concrete system architecture. The design of the logical components helps to check different technical solution implementations against each other and to select the best one without having to make large additional expenditures. This will aid the purpose of the logical viewpoint to provide a more comprehensive consideration of potential realization options and ultimately a better technical architecture.

Technical Viewpoint

The technical viewpoint represents the concluding viewpoint of the architecture framework and the solution space. The content of the viewpoint is the technical solution and its relationship to each other. The term technical solution is specified below.

Technical Solution

The technical solution describes the individual elements and the relationships between those elements, which form the architecture of a system, realize the specified requirements, and provide the necessary functions needed to achieve the system goal(s).

All elements defined in the previous viewpoints converge in the technical viewpoint and form the technical solution of the system of interest. Architectural decisions are based on the technical definition of the logical implementation-neutral components. The content can then be used in subsequent steps for detailed design and finally implementation of the system in its field of operation.

5.4.2.3 Granularity Layers of Core Architecture Framework Concept

After consideration of viewpoints and allocation to problem and solution space the possibility to detail content within the different areas of the architecture framework shall be considered. The concept of granularity layers is introduced and shown with respect to the defined viewpoints in Figure 37. The term granularity itself has already been defined in section 2.1.2 and was put in relation to abstraction in section 0.

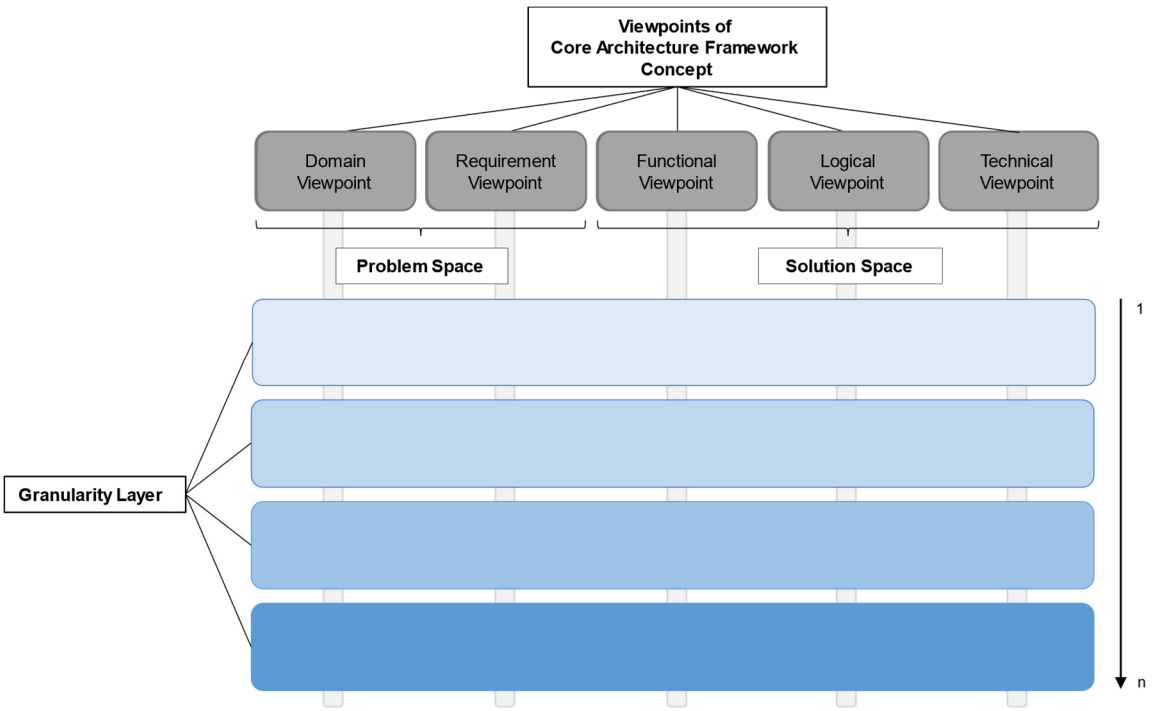


Figure 37: Granularity layers of core architecture framework concept

Since granularity of a system can be regarded as fluid without a clear separation and is strongly dependent on the mindset and professional background of the creator of an architecture description, it is attempted to force the user to break down complex issues into different layers of detail within the architecture framework concept. The reason is to break down complex considerations in to smaller and manageable parts to enhance the problem solving and specification of solutions. This shall be achieved by the introduction of detail levels, so-called “Granularity Layers”. This should lead to a consideration of a clearly defined set of elements, simplify the design process, and make process less error prone. Nevertheless, it should be mentioned that a strict rule for classification of specific elements to specific granularity layers is not suitable due to the individuality of systems considered within the architecture framework concept. Therefore, the actual assignment of content still depends on the executing stakeholder and their creativity, which shall be triggered by the provided granularity layers. A possible general granulation of production systems that can be used with a suitable system is shown in [5]. The given structure of the granularity layers can thus be used as an instrument to break up content and finally to simplify the application of the architecture framework concept methodologies considered in the micro cycles.

As already shown in Figure 35, granularity is distributed vertically, so the content is spread vertically from very rough to very detailed as well, which equals a low level of granularity on the top layer of the core architecture framework concept and a high level of granularity on the bottom layer. Those granularity considerations also apply to the

individual viewpoints as graphically shown in Figure 37. In order to achieve an efficient and solution-oriented architecture definition, the contents of the different viewpoints should be selected in such a way that they correspond and can be related to each other in their layer of granularity (see section 5.4.3.3). The number of necessary granularity layers (1...n) depends on the individual system(s) and must be defined by the stakeholder applying the core architecture framework concept. Therefore, a fixed number of layers is not feasible and not further specified.

5.4.2.4 Views and Content of Views of Core Architecture Framework Concept

Based on the structure and granularity layers of the architecture framework concept for the introduced viewpoints, so-called "architecture framework views" are defined (view definition see section 3.1.1.2). The view is expressing the architecture from a specific viewpoint and layer of granularity ([34] based on [30]). Depending on the viewpoint, the focus on different architectural contents of the system of the individual views is limited to the content represented within the related viewpoint. The individual views considered relevant within the core architecture framework concept are derived based on literature, the experiences made within the research project CrESt, and the contents of the previous chapters. A general procedure for developing views is described by [123]. The allocation of views to the structure of the core architecture framework concept is exemplarily depicted in Figure 38.

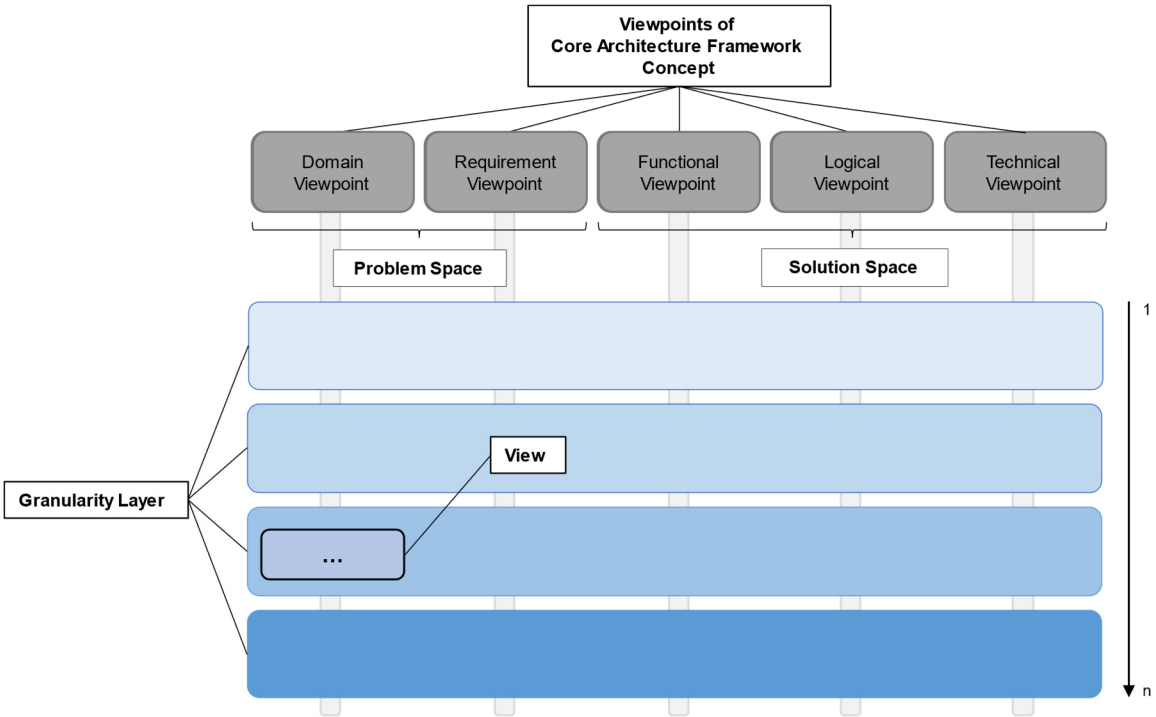


Figure 38: Views within core architecture framework concept

The utilization of additional views may be relevant depending on the type of system of interest but are not considered in the following. In the context of this thesis the consideration of views is limited to the ones shown in Figure 40 and documented below. If the use of additional views outside of this thesis shall be considered, it has to be checked to what extent the core architecture framework concept has to be adapted regarding structure (e.g., additional viewpoints), method, and content types. In the following, the specified views are introduced and briefly described.

Views of Domain Viewpoint

Within the domain viewpoint the domain and the context of a system of interest are considered at different layers of granularity. The “Domain View” and the “Context View” are used for this purpose. The views within the domain viewpoint shall be utilized to define the environment of the system, the exact delimitation of the system to this environment, and all relevant elements within (e.g., other systems, stakeholders, and normative requirements like laws). The purpose is to create a first rough specification of the system as a common discussion and communication basis to make sure that all involved disciplines and stakeholders start from a similar point, ideally the same basis. Especially at the beginning of such a definition process it is very important that a common starting point is available to avoid unnecessary iteration processes and to minimize misunderstandings due to different states of knowledge. Therefore, each view considered certain partial aspects, which are part of the overall content. Since the views are located within the same viewpoint, they are directly related to each other in terms of their content. Based on the results of the domain and the context view the content within the views is iteratively detailed.

The domain view is described in detail, as an example, in Table 6. In the table the most important core facts of the considered view are described. Besides name, viewpoint and problem/solution space affiliation, relationship to other views, the goal and purpose of the view are described. Furthermore, the content and the procedure for defining the content is shown. A detailed description of all views in tabular form can be found in the annex (see Table 17 to Table 24).

Table 6: Description of domain view

Name of View	**Domain View**
Related Viewpoint	Domain Viewpoint
Problem / Solution Space	Problem Space
Goal[4] of View	Define all elements within the considered domain with a direct relationship or influence on the system of interest
Purpose[5] of View	The specification of the considered domain and the contained system promotes understanding of the system among relevant stakeholders during the definition process and simplifies following specification steps such as context or requirements definition. Since the stakeholders have a similar common understanding of the environment in which the system is operated, the possibility of misunderstandings due to different knowledge bases is also reduced.
Element Types and Model Kind of View	Element Types: • Domain Element • Stakeholder • Relationships for description of connection between stakeholder(s) and domain element(s) Model kind: • Model of Domain (within tool: Domain Model Diagram)
Procedure(s) within View	First of all, the line between relevant and irrelevant context within the system environment must be drawn. Not all elements in the environment are relevant for the specific system (for definition see section 2.1.3). This distinction is made by the stakeholders and is based mainly on experience and known facts about the system of interest. This boundary may change slightly during the design process as new information becomes available. After the relevant domain has been delimited from the rest of the environment, all relevant stakeholders on the one hand and relevant domain elements (e.g., other subsystems) on the other hand have to be defined. In addition, appropriate relationships between the elements of the domain and the system and among each other are specified. The process of defining elements, stakeholders, and relationships is mostly iterative but depends strongly on the stakeholder conducting the definition. After all elements, stakeholders and relationships are defined, the result is a model of the domain in which the system of interest and the relevant domain is described. This model then serves as the basis for further definition steps such as the context definition. An example of a domain view is shown in Figure 80.

[4] "the act of stating clearly what you want to achieve or what you want someone else to achieve" [167].

[5] "an intention or aim; a reason for doing something or for allowing something to happen" [168].

Name of View	Domain View
Relationship to other Views within Viewpoint	Domain view (on different granularity layer) and context view,
Relationship to other Views of other Viewpoints	Stakeholder need view, use case view, product view, and requirement view

The meta-model of the view, which is representing the element types used for modelling the content of the view, is shown in Figure 39. In addition to the element types “stakeholder” and “domain element” (as described in the table above), pre-specified relationships are defined to connect the element types. The element types can be instantiated for the creation of concrete domain content within the application of the core architecture framework concept. For example, the instances operator and production planner of type stakeholder could be defined, which have a dependency relationship to the instance production system of type domain element. The meta-models of the other views as well as an allocation of the views to each other is shown in the annex.

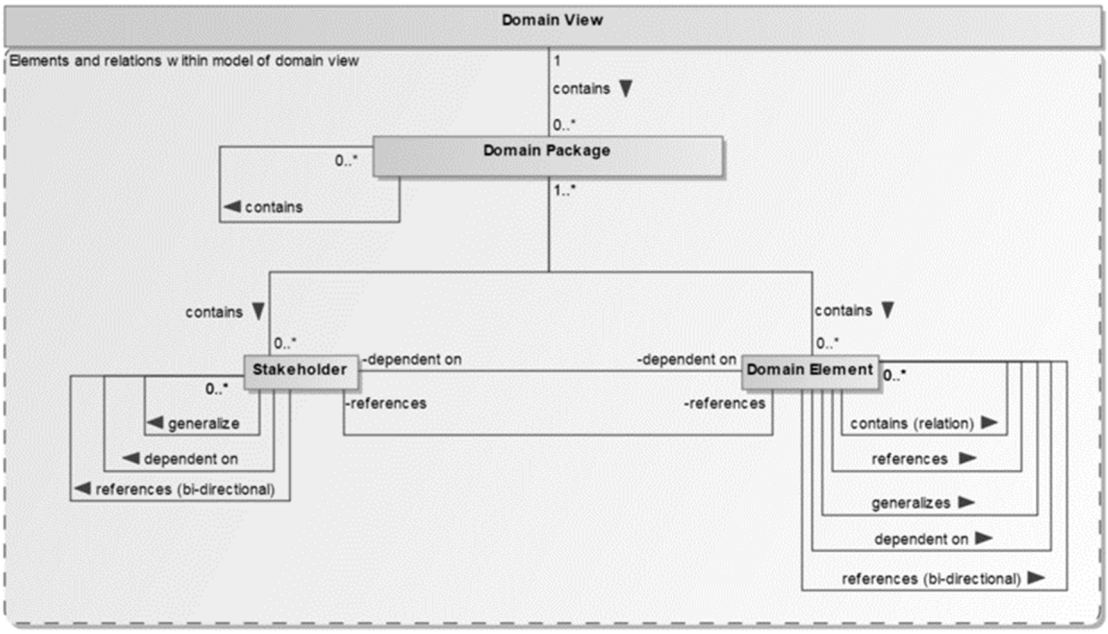


Figure 39: Meta model of domain view [169]

Views of Requirement Viewpoint

Within the requirement viewpoints, four different views are considered per granularity layer, namely the “Stakeholder Need View”, the “Use Case View”, the “Product View”, and the “Requirement View”. The goal of using these views is to describe and define

the problem space and finally the requirements towards the system of interest. The purpose is to detail all requirements, constraints, and qualities resulting from the objectives and tasks of the stakeholders, use cases, and the product. The contents of those views represent parts of the discipline of requirements engineering (as specified by [165]), whose actions sustainably affects the solution design of the system. Only with a clear and complete definition of the requirements, a value-added system can be created that functions smoothly in operation. Since the development of complex systems involves several different stakeholders and disciplines, the starting point must be clearly defined for all of them, so that the different stakeholders/disciplines can carry out their intended task, which shall ultimately resolve in a good match of relevant results. In order to provide such a clear starting point, the views specified above could be assembled in different orders depending on the argumentation and knowledge background of the executing stakeholder. However, within the core architecture framework concept, first the stakeholders, their tasks, and corresponding interest are considered (exemplarily show in Figure 54). Absolutely relevant stakeholders, who should be considered are, for example, defined by [30]. The stakeholder view is followed by the use cases and the product view, which can logically exist in parallel. In the use case view all possible application scenarios that may occur during the life cycle of the system are considered. The product view represents the product(s) and the associated production processes used to manufacture the product within the system of interest. Relations between product and production system are shown on a general level by [5]. By considering different stakeholders, use cases, and product(s) as well as production processes it shall be ensured that the requirements are determined as completely as possible in the requirement view and that it is, for example, avoided that potentially needed system capabilities are not available during operation. In the requirement view the functional requirements, qualities and constraints are defined. This set of requirements is then used to create the relevant architectures and, at each moment of development, to check the current state of the system against the defined requirements in order to evaluate the current state of development (as described, for example, in the V-model – see Figure 15). The detailed description and meta models of the views of the requirement viewpoint, as shown for the domain view, can be found in the annex and in Table 18 to Table 21.

View of Functional Viewpoint

Within the functional viewpoint and the functional views, across the granularity layers, the different functional solutions are considered. The transition between views of the requirements viewpoint and views of the functional viewpoint describes the imaginary

boundary between problem and solution space. The goal of using the functional views is to define functional solutions (functions) for all previously defined requirements, considering qualities and constraints. The entirety of the defined functions is represented within the functional (reference) architecture (depending on application of core architecture framework concept to group of systems or single system). By defining the necessary functional scope of the system of interest, the following architecting steps should be simplified. Based on the functional architecture, it is considered which logical and finally technical solution elements are needed for the system to provide the necessary functions and thus fulfill the requirements. The detailed description of the functional view and meta model can be found in the annex and in Table 22.

View of Logical Viewpoint

Within the logical views logical elements are considered at different granularity layers of the logical viewpoint. A logical element represents a solution that is close to implementation but technically independent. The logical element fulfills one or more functions and requirements. The goal of using logical views is to define all necessary logical components in different levels of detail. The purpose of the application is to reduce the big abstract step between functional solution and technical solution and to create an additional iteration level. This facilitates the traceability of the solution creation. Because a logical element can have several possible technical solutions, different solution possibilities can be defined, evaluated and the most suitable one can be selected before implementing a final technical architecture. This is done in an iterative process between logical and technical viewpoint/views. However, this possibility only becomes available through the consideration of the logical elements within these views. With appropriate documentation of the interrelationships of logical components and possible technical solutions, the application of, e.g., a reference architecture, which is to serve as a template for the creation of a system architecture, is facilitated in the long run, since several possible solutions are available. The detailed description of the logical view and meta model can be found in the annex and in Table 23.

View of Technical Viewpoint

The final definition of the technical solution is made in the technical views. The goal is to define all necessary technical elements and their interrelationships that are necessary to provide a certain performance/service, so that the defined requirements,

qualities, and constraints are met. The purpose of the views is to present the solution as detailed and comprehensible as possible so that they can be used as a basis for all further steps of the following detailed planning and implementation of the system. The results provide the basis for the further design of the system of interest by the individual disciplines. The detailed description of the technical view and meta model can be found in the annex and in Table 24.

In the following, within Figure 40 the views described above as well as the distribution over the viewpoints and along the granularity layers are shown. The figure illustrates the classification of the views in the core architecture framework concept. The dashed boxes represent the possibility of creating additional views, which correspond either to the views already described or to new additional views, for example in the case of an extension of the architecture framework. It should be noted that there can be only one view of the same type on a granularity layer.

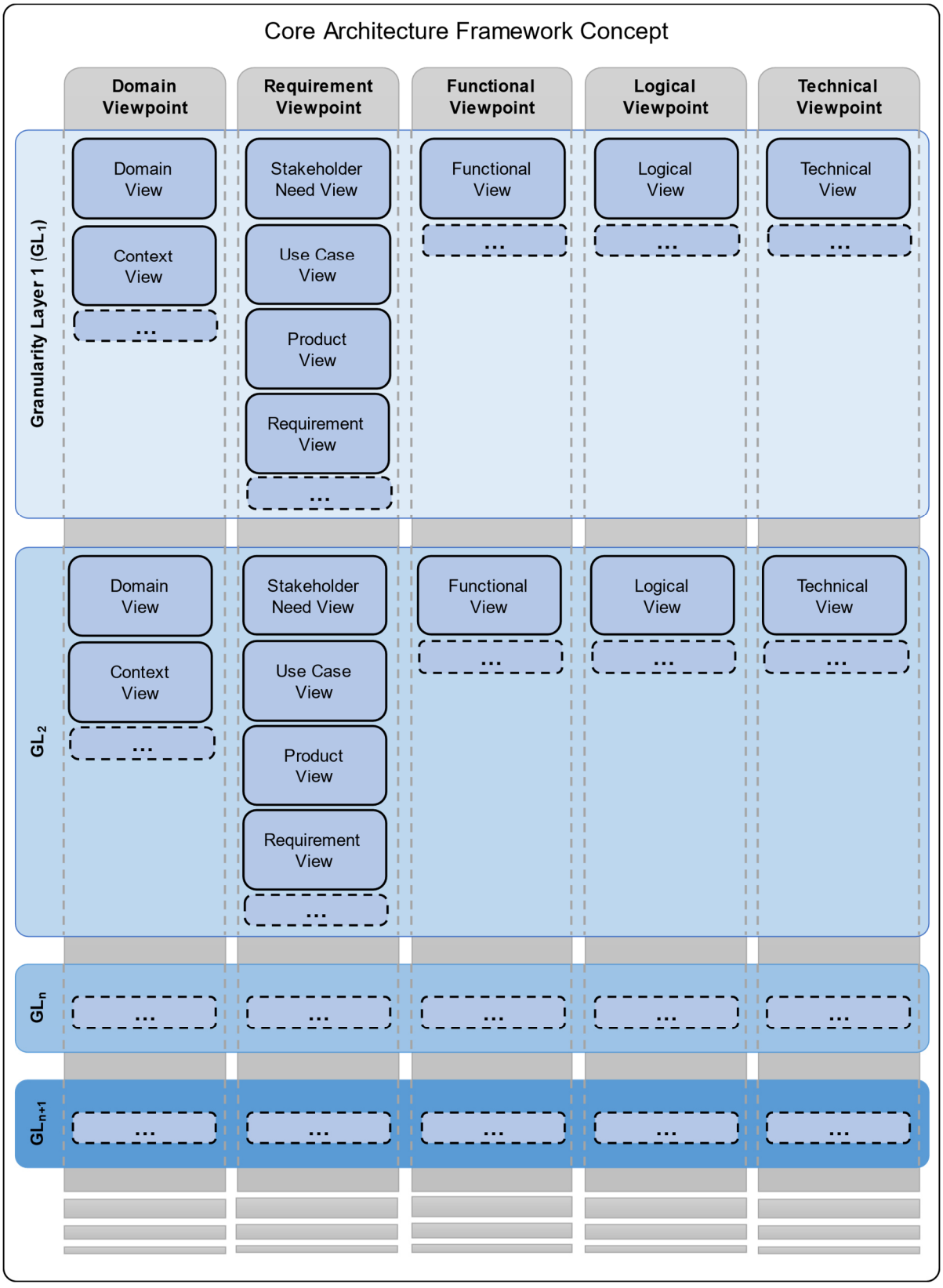


Figure 40: Structure of Core Architecture Framework Concept

Production System Application Example

The structure and considered contents of the architecture framework concept are described in more detail by using the example of the cylinder head manufacturing system. Using common hierarchization concepts for manufacturing companies, for example as presented in [5], which propose hierarchies from the production network (highest level) to construction elements (lowest level), the cylinder head manufacturing system example can be classified to at a level of abstraction that corresponds to a work unit. Assuming that the cylinder head manufacturing system is detailed on two granularity layers only, granularity layer one is considering the work unit, namely the cylinder head manufacturing system and its main components like manufacturing and transportation system, and granularity layer two is considering the work stations, which would represent the single stations within the cylinder head manufacturing system such as the milling and the assembly system (Figure 10). The corresponding view contents are detailed accordingly. For example, the functional view of the cylinder head manufacturing system on GL1 considers functions that are clustered into groups of functions like "Handle & process information", "Plan manufacturing", and "Operate production system". As an example for detailing of functions, within the function group "Operate production system" functions on GL1 like “transport material within the production system (1)” or “produce product portfolio (2)” are considered. Those functions are detailed on GL2 into functions like “distribute/transport material within the (sub-)system (1.1)”, “handle square/ cylindrical products (1.2)”, “mill cylinder head contour (2.1)”, “grind cylinder head (2.2)”, and “assemble cylinder head (2.3)”.

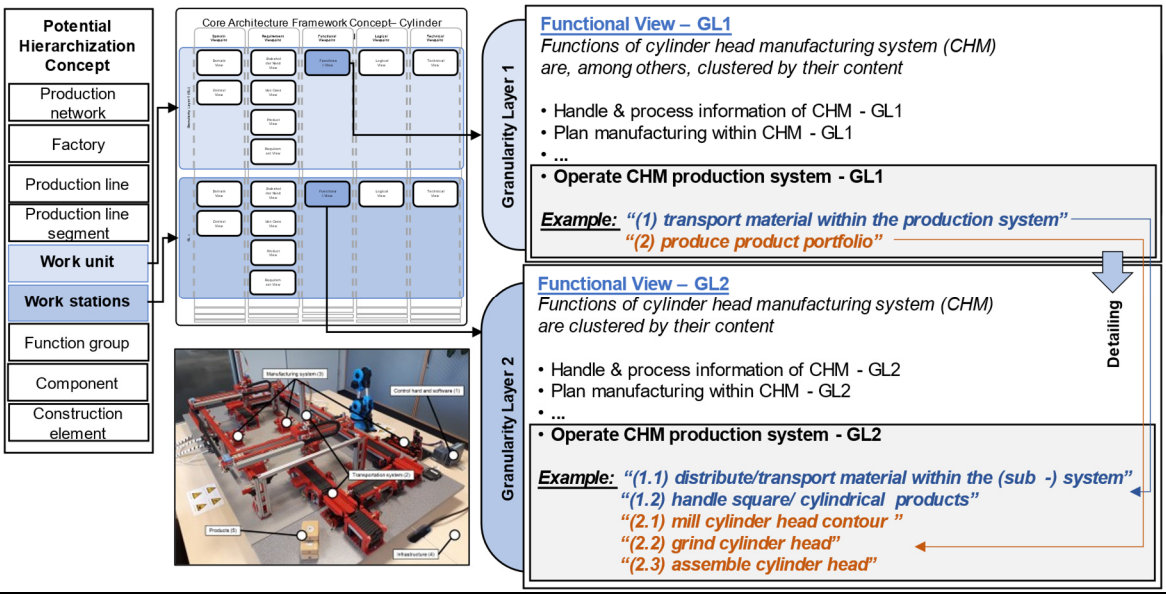


Figure 41: Production system application example – structure and content of core architecture framework concept

5.4.2.5 Simplified Representation of Architecture Framework Concept

After the structure and content of the core architecture framework concept for the definition of a system or reference architecture have been specified, the next section will consider the necessary methodologies (micro cycles). For further considerations of the introduced concept in this thesis a simplified illustration of the core architecture framework concept will be presented, as the illustrations of the detailed structure of the core architecture framework concept can quickly become too complex in the context of, for example, the methodological considerations. The illustration in Figure 42 is reduced to the two-essential structure-giving concepts viewpoints and granularity layers (three granularity layers are exemplarily shown for the simplified representation). Based on those two concepts, the resulting views on the specific granularity layers are represented by boxes. Those simplifications and the resulting representation are shown below in Figure 42. For all following considerations the simplified representation of the core architecture framework concept is used.

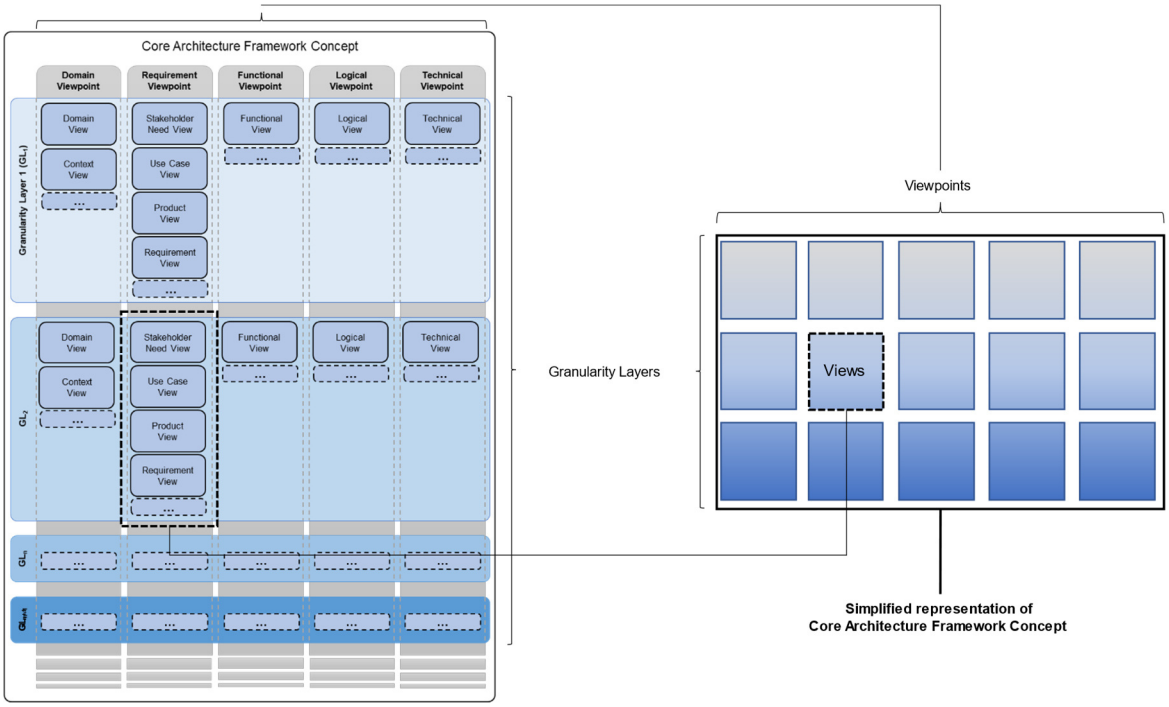


Figure 42: Simplification of architecture framework representation

5.4.3 Micro Cycle - Methods of Core Architecture Framework Concept

In addition to the overall procedure represented by the macro cycle and within the core architecture framework concept, in the following, the micro cycle concerned with transition as well as creation and change of content within the core architecture framework concept is presented. The micro cycle is represented by different methods,

which connect the structural and content related components of the core architecture framework concept and methodically support the creation and change of content. Before the methodological approach is explained in more detail, a digression on the top-down and bottom-up approach is necessary to determine which basic general concept is pursued within the core architectural framework concept. Thereafter the main methods are introduced, described, and accordingly aligned to the selected approach. On the one hand a methodology for the transfer between views/viewpoints is described. On the other hand, a methodical concept for the definition of contents within the viewpoints/views is characterized and how changes can be taken into account is defined. The whole methodology focuses on the connection of the structural elements and on the processes within these elements. An overview of the methodical consideration within this section is given in Figure 43.

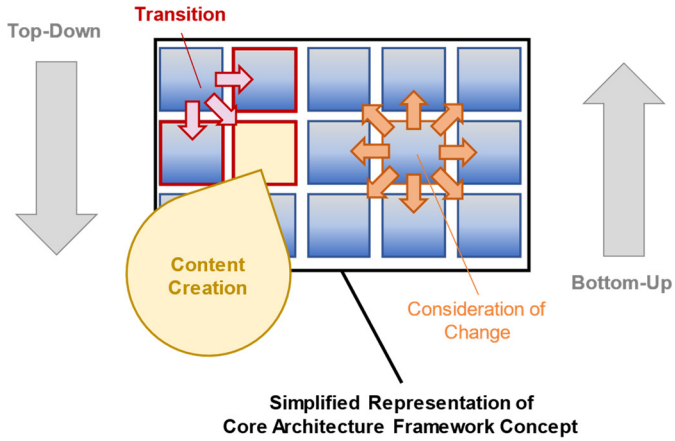


Figure 43: Methodologies considered within core architecture framework concept

5.4.3.1 Evaluation of Top-Down and Bottom-Up Approach

For the realization of complex systems, two main design approaches, the top-down approach and the bottom-up approach, have emerged and established themselves over time [170]. The approaches and possible hybrid forms are not only used in systems engineering but also in other areas of a company and have also proven to increase performance there [171]. Both the top-down and bottom-up concepts describe general procedures for problem solving utilizing fundamentally different starting points. The usability of the two approaches for the definition of architectural content is strongly dependent on the planned application, the situational environment/domain, and the goal which is pursued. Since both approaches can be significant in the context of the further procedure for the development of the (core) architecture framework concept, both approaches are introduced and compared to

each other by considering advantages as well as disadvantages of the approaches within this section.

The top-down approach describes, with respect to a considered system of interest, a problem-solving procedure and design approach later utilized within development or manufacturing of a system [172, 173]. In the top-down approach, all relevant elements are first identified at the top-level of a considered system and thus on the lowest level of granularity [172–174]. The identified elements are subdivided and their structure is completed by adding the subordinated related elements [174]. Within this procedure the considered system element/problem is described step by step in more detail [173]. Through the step-by-step division and detailing on the subordinate level of a considered problem/system element, one moves during the processing from top to bottom, i.e. from the lowest to the highest granularity layer [172–174]. The top-down approach ends when the highest relevant layer of granularity has been reached and the system of interest has been broken down into its constituent parts [172–174]. The layer of granularity at which the top-down approach is completed strongly depends on the problem/system of interest and the experience-based assessment of the stakeholder(s) involved. The procedure must be applied until the results are sufficient for the next processing step, for example, the detailed engineering of a system. In short, the top-down approach for the development of a system of interest can be defined as follows.

Top-Down Approach

The top-down approach can be defined as “designing a system [...] by identifying its major components, dividing them into their lower level components, and then repeating the process until a designated level of detail is achieved” [174].

Compared to the top-down approach, the bottom-up approach pursues a different, opposing starting point. The procedure first refers to the highest layer of granularity and then considers all remaining layers up to the lowest layer of granularity step by step until a desired goal is achieved [175]. For the design of a production system, this means that the structures and components of the highest granularity layer are considered first [89]. These different elements are then assembled throughout the process from "bottom" to "top” into more complex solutions until the design of the system of interest is complete and thus finalized [89, 175, 176]. An additional comprehensive description of the procedure can be found in [72]. In summary, the creation of a system in the bottom-up approach can be defined as follows.

Bottom-Up Approach

The bottom-up approach is used to design systems "[...] by starting with the most basic or primitive components and proceeding to higher-level components or modules by using the lower-level tested and approved components as building blocks, until the system design is completed" [177].

Both approaches have advantages and disadvantages, which are shortly summarized in the following.

In the context of a model/simulation-based application of bottom-up approach, there are advantages in the error correction itself, since errors usually occur at the currently considered level due to the already completed and verified components at the subordinate level. However, this has a disadvantage with regard to the overall design, since this can only be determined and verified at the top layer and, in the worst case, during a possible implementation. This leads to unplanned efforts and thus costs that are significantly higher than planned. In addition, the technology dependency defined from the bottom granularity layer can have a negative impact on the design when new or changed technologies appear. This could lead to the need of revising large parts of the overall design. [89]

Compared to the bottom-up approach, possible design weaknesses with regard to the overall system become apparent earlier within the top-down approach, since the necessary behavior or the functions to be provided by the system can be defined and checked in an earlier state of the design. The functional description also defines the scope of a system more clearly and completely than requirements in natural language. Since the process proceeds from the lowest to the highest layer of granularity, i.e., from top to bottom, the design of the system is more technology-neutral and less technology-dependent. This means that different technologies can be used for the realization of a system depending on the goal and use case. A downside of the top-down approach is that defined content on the top layer is used as input for the definition of content on the bottom layer and this can lead to possible planned but technically unfeasible solutions. This would then lead to parts of the design having to be redesigned. [89]

The assessment of which approach is best suited for a specific application depends strongly on the goal and purpose of the system development as well as the actual starting situation. Factors such as number of similar or identical systems already designed, new development or planned system expansion can be relevant in such considerations and must be weighed against the advantages and disadvantages of the

approaches [89]. A combination of the two approaches is also not unlikely in the context of the overall development of a project [89]. This can be seen, for example, in Figure 15 representing the introduced V-model for development of systems. Therein the basic engineering and parts of the detailed engineering are conducted in a top-down manner and the implementation of the system in a bottom-up fashion.

For the (core) architecture framework concept presented in this thesis and in reference to the V-model, a top-down approach is selected as general procedure to which the methods are aligned. The reason for that is mainly the longest possible technology neutrality, which is advantageous to the development of system architectures that take reference architecture content into account. This preserves design freedom on a technical level for a longer period of time and simplifies the derivation and adaptation of content, since the early technological definition is delayed for a longer period of time compared to the bottom-up approach. In addition, the rapid discovery of potential design flaws facilitates the ongoing development of new system architectures based on reference architectures. Furthermore, the existing initial situation, in which not many detailed technical solutions are available, restricts the choice of approach. The disadvantages of the top-down approach are primarily content-related issues that are not as important in the context of framework development as, for example, a methodological influence.

5.4.3.2 Generally Possible Procedures for the Development of a System

As a preliminary work for the methodologies that determine which contents can be considered "where" and "how" in the core architecture framework concept, three general procedures are defined in the following and shown in Figure 44. Those procedures can be applied in the selected top-down approach as well as in the bottom-up approach (neglected in this thesis). The first procedure considered a granularity layer focused definition. In that procedure, the views of the different viewpoints are run through at the same layer of granularity before the definition is continued at a more detailed layer in the same way. The second procedure describes a viewpoint focused definition. Within that procedure all views of a single viewpoint are run through one after the other before the transition to the next viewpoint takes place. For the first two options, it is assumed that the procedure will be carried out as prescribed and all contents will be defined one by one until a complete architecture description of a system is created as a result. Deviations or omitting steps from the procedure is not accepted. As such a strict procedure is not always possible in a company or an engineering project due to, for example, the processing of the project by different disciplines, a third option closer to reality is considered. The third procedure is a mixed

definition approach. The third option describes a mixture of the first two procedures for the definition of an architecture. All three procedures are valid for both a top-down and a bottom-up approach and are shown in Figure 44.

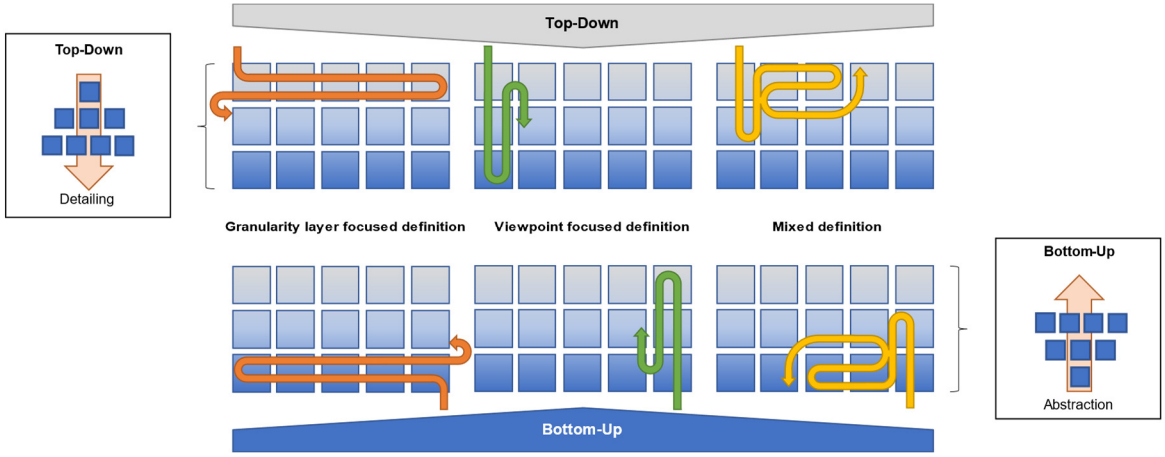


Figure 44: Procedures for defining content within architecture framework

For procedure three, the mixed definition, the strict definition sequence of the first two options is not followed, and an interruption of one procedure and the continuation of another is allowed. This is exemplarily shown in Figure 45. For example, a granularity layer focused approach could be followed first (first 3 viewpoints - orange arrow) and then a viewpoint focused approach (first viewpoint - green arrow). Despite this less strict approach, option 3 does not allow to skip or completely omit elements that are logically consecutive and necessary to define (red dotted arrow). This would lead to difficulties regarding the methodological support and it would be much more difficult for other disciplines or stakeholders to review and comprehend made decisions as well as the completeness of results. For the made example the following picture emerges in Figure 45. After the partial horizontal as well as vertical definition, the elements marked with green X's can be defined next, if either a granularity layer or viewpoint focused approach is further pursued. Content of views covered by the green and orange arrow are considered to be fully defined. The suggested individual sequence of procedure one and two is not changed by the mix of procedures and must be followed up. The fields marked with a red minus can only be filled with content when they are logically next in line when applying of the two procedure variants.

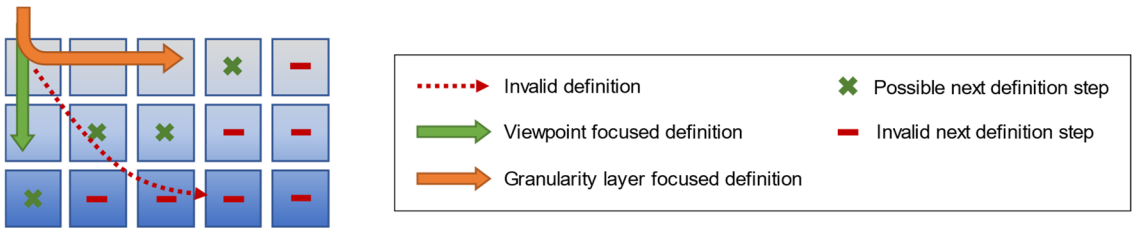


Figure 45: Example of mixed definition

Based on those general considerations and assumptions, more detailed preconditions and rules for transition between views and defining content are described in the following sections.

5.4.3.3 View Validity Determination Method

Based on the mixed definition procedure the following section defines rules and a method for the determination of the validity of the use of a selected view during the creation of architecture content. The question of the validity for using another view always arises when a shift is made from an already defined view to another to be defined view. A shift between viewpoints always occurs when a switch happens between views that are located in different viewpoints. Since the content of a viewpoint is represented in the form of views, in the following it will only be referred to as the shift between views. The shift between viewpoints is implied and not mentioned explicitly each time. The goal of this section is to clearly define in which cases, and under which preconditions a shift between views within a top-down approach for the definition of architecture content is allowed. The purpose is to make the shift between views and affected viewpoints comprehensible, traceable, and the overall result less error prone. In addition, a regulated shift forms the basis for the definition of content within the views. It thus forms an important precondition and determines when content can or cannot be defined.

As a basis for the definition of the procedure and the necessary rules, in a first step, the dependencies between the individual views are considered. As described in section 5.4.2.4, there is a relationship between the contents of the various views. This relationship between the contents of the individual views and the applied top-down approach implies that for a complete description of an architecture, certain contents should be present before other contents. For example, the definition of the logical elements should not take place ahead of the definition of the relevant stakeholders or system requirements (at the same granularity layer).

When considering the definition of the functional view on granularity layer 2, shown in Figure 46 as an example, it can be seen that when following a granularity layer focused

procedure, a definition of all views within the defined viewpoints on granularity layer 1 as well as the upstream views on GL2 has to be carried out before the definition of the functional view on GL2. Following a viewpoint focused procedure, all views within the first two viewpoints across all granularity layers and the superordinate views within the functional viewpoint can be defined before the exemplary selected functional view. Both procedures assumed that all views within the viewpoints and at the different granularity layers are run through and defined one after the other (as shown in Figure 44). In reality, this is not always possible because of the workflows in a company and the processing of an architecture definition across several disciplines. Therefore, it is assumed as shown in section 5.4.3.1 that a mixed form of the two procedures is applicable under the restriction that no arbitrary mix may arise and that the skipping of logically sequential definition steps is forbidden. For the reasons mentioned above, a mixed definition procedure is assumed as standard within the core architecture framework concept.

Based on the three procedures previously described, the different potentially defined and mandatory contents for the definition of view content can be derived. As shown in Figure 46, when procedure one and two are considered separately, the defined views (orange and green squares) are specified before the currently processed view is defined (blue square). When combining the two procedures shown in Figure 46, for procedure ”mixed definition”, the situation arises, within a top-down approach, that certain views are required for the definition of the currently process square (black dots) and other views can be potentially already defined, but are not necessarily required (framed squares in orange and green).

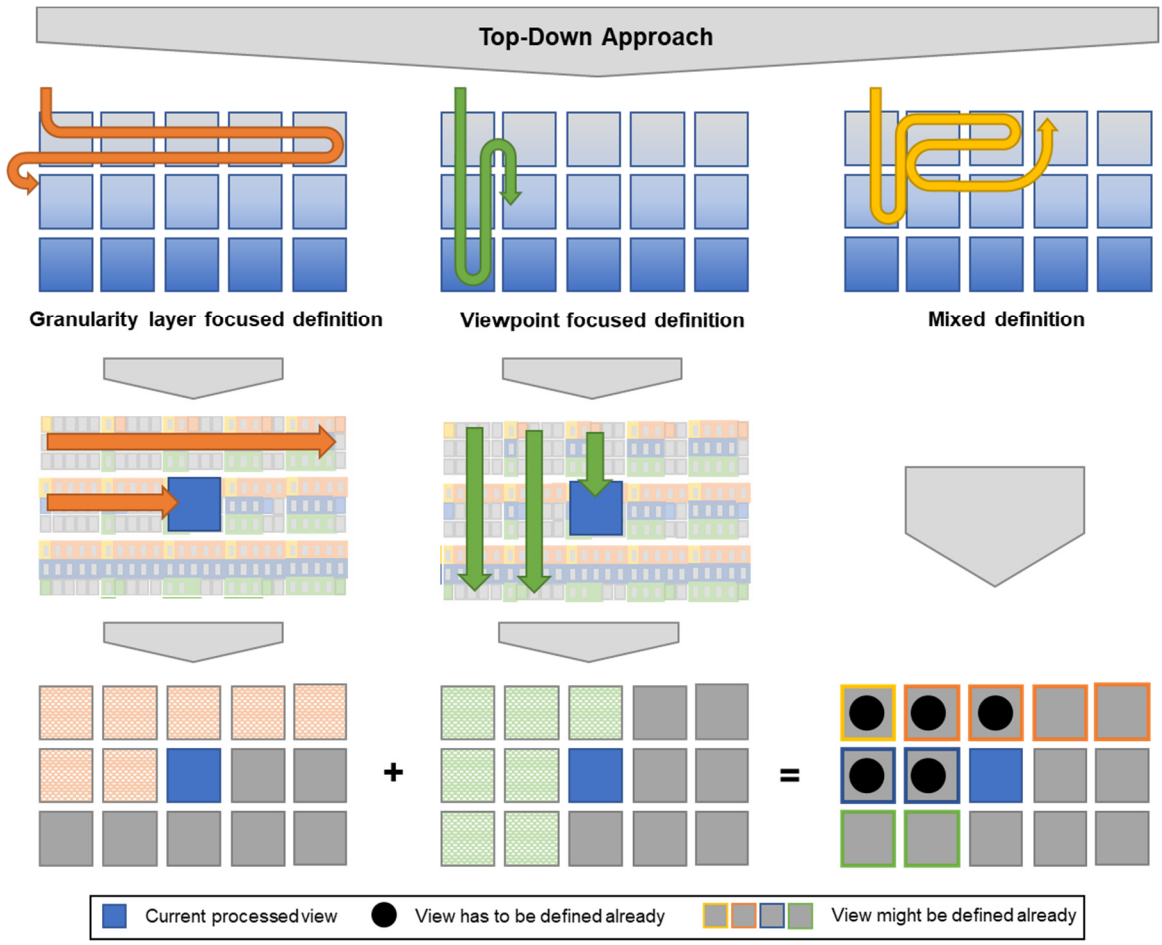


Figure 46: Specification of available contents during application of core architecture framework concept

Following Figure 46 and with the help of the simplified representation of the core architecture framework concept, the different conditions of content dependency when considering different views during the definition of an architectural description are shown in Figure 47. This representation might be misleading, when assuming that only one architecture description shall be created. Of course, only one core architecture framework concept is applied for the definition of content, but the figure shows the different states of the definition of views at different points in time. For this reason, the simplified representation of the core architecture framework concept is shown several times next to each other representing all the different contents of the views needed to compose the overall architecture description of the system of interest. The general flow direction for the top-down definition extends from the views within the domain viewpoint at the lowest granularity layer (GL1) to the views of the technical viewpoint at the highest granularity layer (see Figure 47).

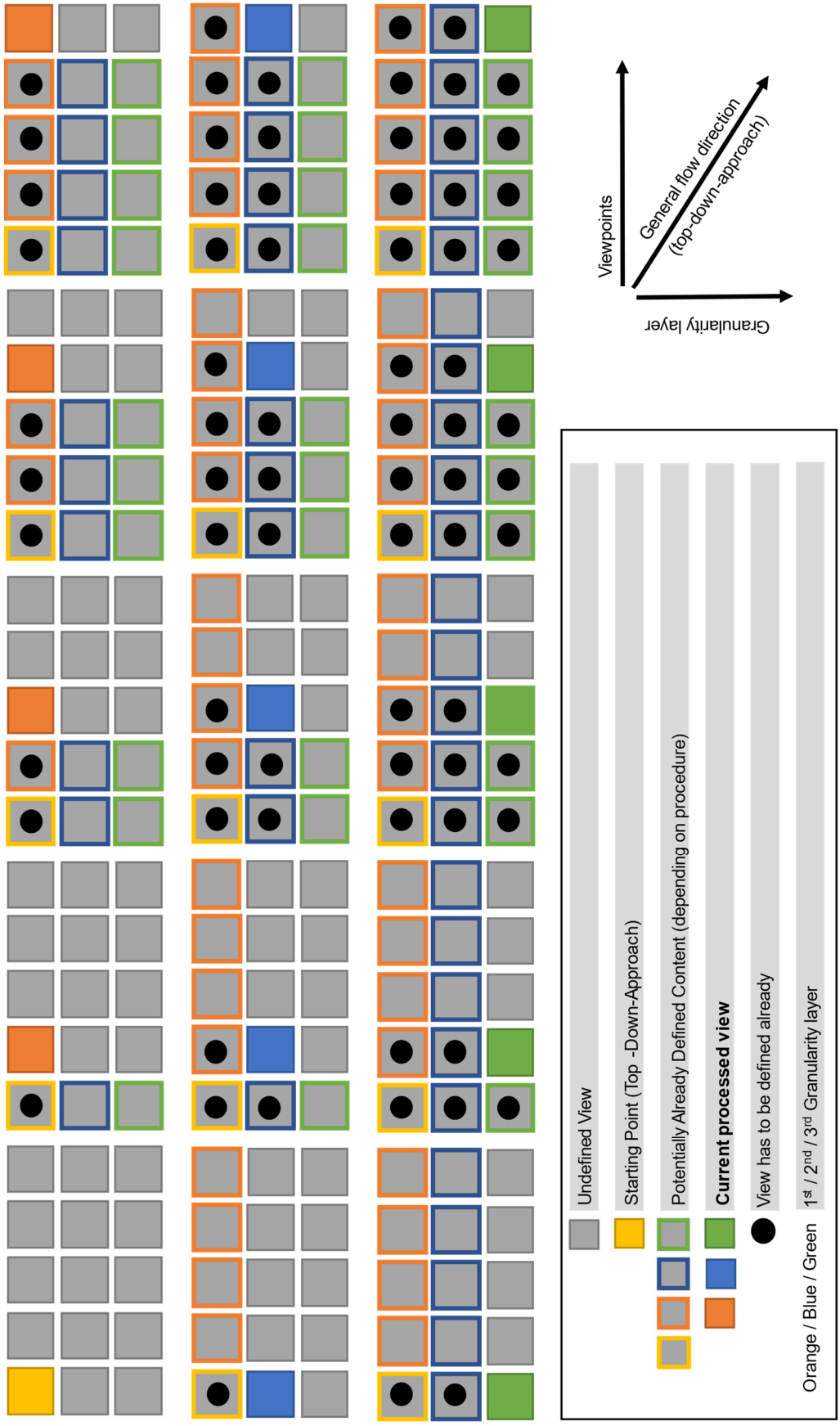


Figure 47: All possible states of the core architectural framework concept during the definition of an architectural description.

In the figure, a distinction is made between filled, framed, and unchecked squares. The squares that are not highlighted with color describe undefined views (gray squares). The framed squares express that when following a top-down approach it is possible that the content of a view has already been defined. The squares additionally filled with a black dot are required for the at this time considered definition of the view of interest (filled square). If the content of a superordinate view or an upstream view has been defined, strongly depends on the applied procedure (see Figure 44). The color-coding orange, blue, and green represents the three granularity layers chosen as examples (GL1 in orange, GL2 in blue, and GL3 in green). The starting point in the architecture framework concept taking the followed top-down approach into account is marked in yellow.

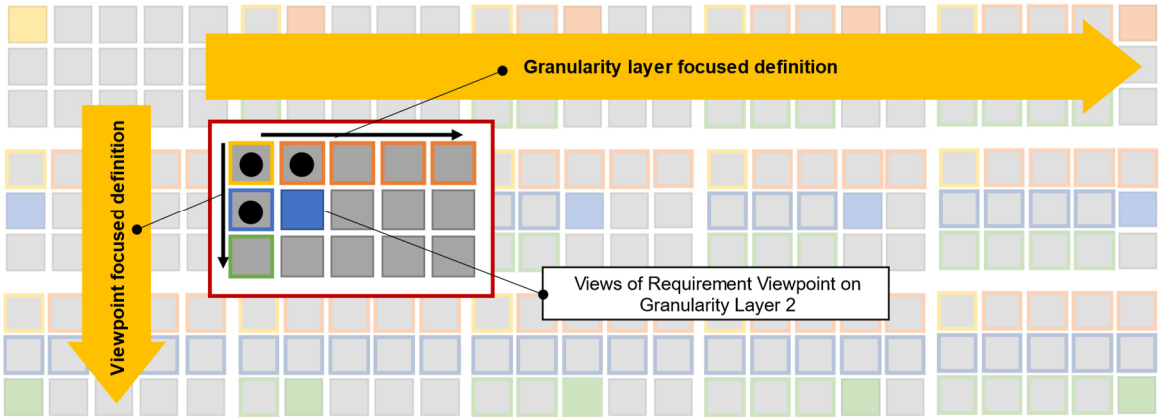


Figure 48: Dependencies and mandatory inputs for the definition of view content

When looking at the example in Figure 48, the representation of seven different views (framed squares) can be defined within different viewpoints before defining the exemplary view pf interest (blue square). Some squares represent as shown in Figure 40 more than one view. For the definition of the exemplary considered views of the requirement viewpoint on GL2, only the views with a black dot are considered as required. On the one hand, the superordinate views of the same viewpoint, whose contents are detailed and serve as input for the view of interest currently considered, are mandatory, in this example the view(s) of the requirement viewpoint on GL1 and GL2 (depending on the specific view considered on GL2). On the other hand, the view(s) of the upstream viewpoint on the same granularity layer are compulsory as well. Those views are the views of the domain viewpoint on GL2. At this moment, all other views (framed squares) within upstream viewpoints or superordinate granularity layers have a rather indirect influence that is not directly quantifiable. These inputs

only serve as additional orientation for the architect and potentially enable a more holistic view on the system of interest.

In the following, based on these determinations, preconditions for a definition of content can be derived, rules for the shift between views specified, and exceptions for special cases made.

Rules for the shift between views within core architecture framework concept

Based on the previous considerations, it is determined that the shift between views with content already defined to views without defined content within the architecture framework concept takes place exclusively

1) horizontally to views at the same granularity layer in directly neighboring viewpoint or
2) vertically to views at a directly neighboring and subordinate level within the same viewpoint.

The general shift or the change between views, in order to define architecture contents, can only take place under certain preconditions. In principle, the rules for a shift already correspond to the general top-down definition flow previously described but must still comply with the rules that content may not be skipped or completely omitted. Therefore, certain preconditions regarding the required input for the planned definition of contents are specified. Those must be fulfilled, so that the shift can be accomplished and ultimately the entire system can be defined.

Preconditions for the definition of content of views

From the consideration above it can generally be deduced that the definition of content elements within a view is only possible if,

1) the relevant elements of the view(s) on the same granularity layer of the upstream viewpoint are completely specified and available, and
2) the relevant elements in the higher-level view(s) of the same viewpoint at the same and directly neighboring overarching granularity layers are fully defined and available.

Both conditions must be met to define architecture content within a view. The preconditions apply to all views within the architecture framework concept. In Figure 49 a potential example and the application of potential shifts as well as required preconditions are shown. In order to fulfill the defined rules for the shift between

views, there must always be vertical input for a horizontal shift (granularity layer focused definition) and always be horizontal input for vertical shift (viewpoint focused definition). Indirect input can also play a role in addition to direct input, although the content is not considered directly but indirectly via the views that provide the direct input.

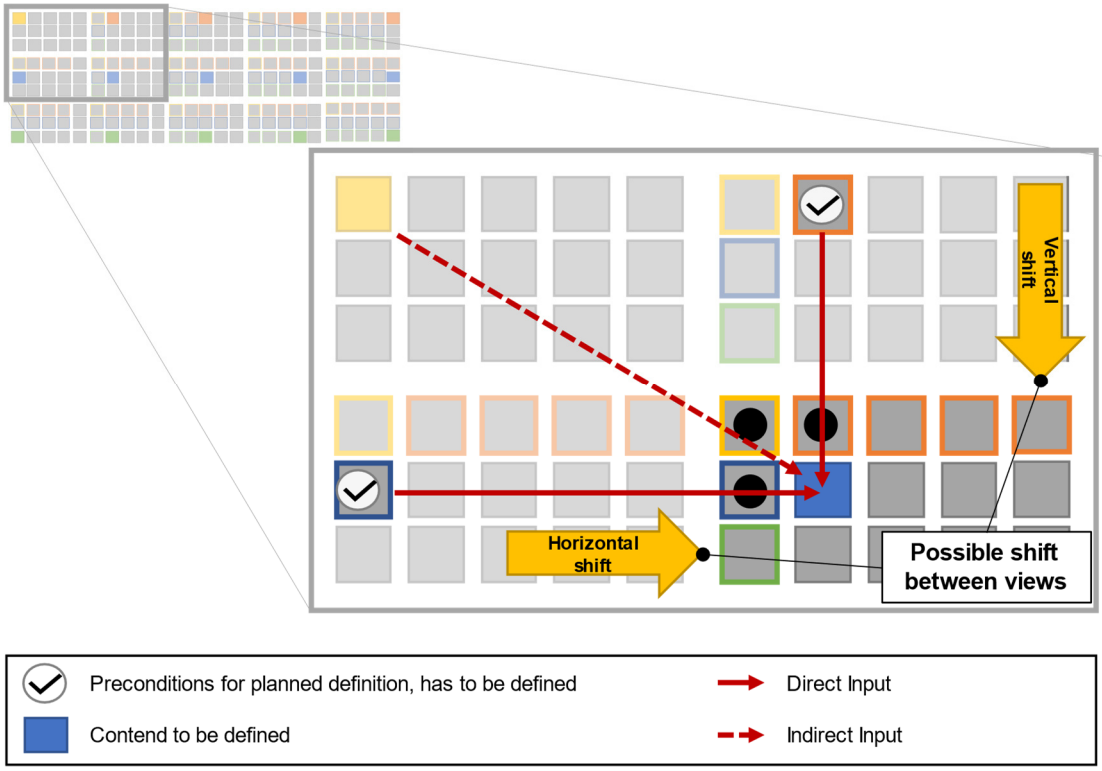


Figure 49: Application of preconditions and transition rules

Exceptions arise due to the structure of the architecture framework concept. No vertical input is available for horizontal movement on the lowest granularity layer, as no higher-level content is available due to the predefined framework structure (granularity layer focused definition). Another special case is represented by the views of the first viewpoint on the lowest granularity layer (GL1), since these have neither vertical nor horizontal input in the architecture framework concept. For the exceptions described, the preconditions continue to apply with the restriction that some inputs are not mandatory because they cannot be provided due to the structural design of the core architecture framework concept. The procedure for determining the validity of the selected view of interest for the creation of architectural content, which can be derived from the basic considerations described above, is shown in Figure 50. The procedure includes that when starting from any defined view a view not yet defined shall be pre-selected by the stakeholder based on the introduced rules for a shift. In a second step,

the preselected view shall be checked, whether the defined precondition for the definition of content within the preselected view is possible or not. If this is not the case and an invalid view has been selected, the procedure can be started again and another view can be preselected. If a valid view has been selected, the definition of the content continues. The corresponding procedure for creating content in connection with the application of the architecture framework concept is described in the following sections.

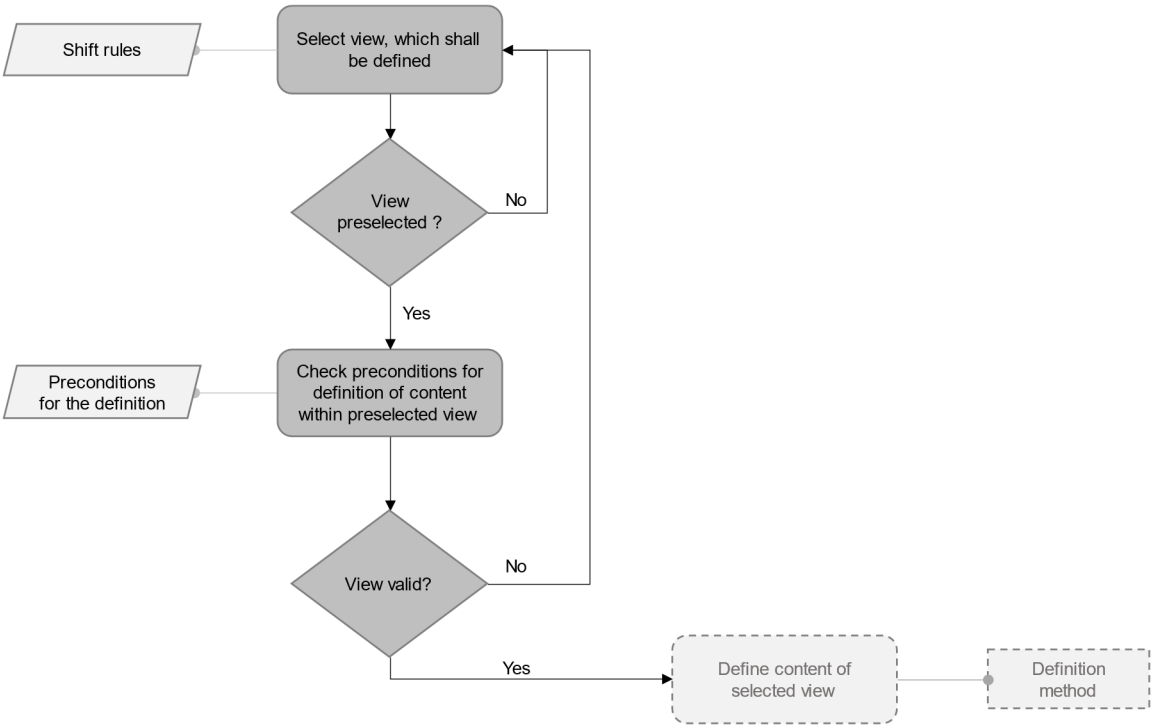


Figure 50: View validity determination method

5.4.3.4 Definition of Elements within Views

After the structure, the content (element types) as well as the shift between the views and viewpoints of the core architecture framework concept have been defined in the previous sections, this chapter concludes with the description of the methodical procedure for the creation as well as adaption of relevant content. Since the creation of content when applying the architecture framework concept is a recurring process, a general procedure is presented Figure 51 that can be applied in a repeatable manner for the different views. The individual design arises from the intended and content described in sections 5.4.2.2/5.4.2.4, addressing the individual viewpoints and ultimately the views contained therein. The presented definition procedure can be structurally limited to three main tasks as well as to a procedure that describes how

and in which order the tasks are passed through to create an architecture description. Figure 51 shows these tasks, which shall be considered for the definition of content within a view. Since the application of the definition procedure is intended to be valid for all views within the architecture framework, the three main tasks are kept general deliberately but describe the core content of each view. In each view, the first task is to consider all relevant inputs. Those inputs (green arrows in Figure 51) emerge from upstream and superordinate views (described in section 5.4.3.3). Following the general flow direction (blue arrow) and based on that input, a process takes place in which relevant architectural content is created with reference to the view of interest. Finally, the results are evaluated accordingly, and a decision is made whether the result is sufficient for the use within other views or not.

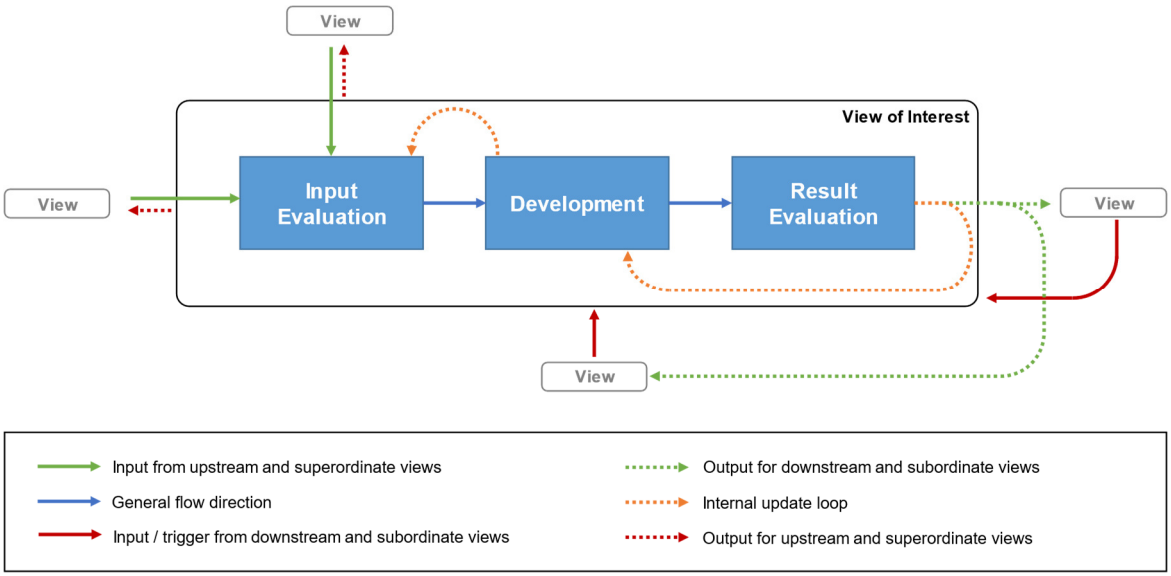


Figure 51: Main tasks for the definition of view content

If the results are insufficient the process can be applied repeatedly until the results are satisfactory from the point of view of the applying stakeholder (orange dotted arrows). As soon as results are evaluated and deemed sufficient, they can be used in another view (green dotted line). After a successful shift, which complies with the established rules, the same tasks arise for the new view of interest. The three main tasks are therefore generally referred to in the following as input evaluation, development, and result evaluation. In addition, it should be mentioned, that an iterative rework of created contents might be triggered from downstream and subordinate views coursed by relevant changes (red arrows). Even though the input might be different, the overall procedure of the architecture framework concept is designed in a way that it can also be applied without adaptions of the methods. The

change of content within the view of interest might also generate an output relevant for upstream and superordinate views in terms of overall consistency of content.

The procedure for considering change is described in 5.4.3.5 in more detail. In the following, the main tasks described are linked together by an iterative process, which guides the applying stakeholder through the tasks. In general, this process is shown in Figure 52 and in connection with the main tasks in Figure 53.

Since development is not linear but often dynamic due to the constantly changing environment and the resulting influences on a system of interest, the process for the architecture framework concept should be continuously and repeatedly applied [77]. Therefore, a general process inspired by quality improvement procedures like the Plan-Do-Check-Act Cycle [178, 179], verification and validation process like [90], and problem-solving concepts like those introduced in Pahl/Beitz [77] and VDI 2206 [25] based on [180] is defined. The iterative procedure consists of the three steps “scope & adapt”, “develop”, and “check”.

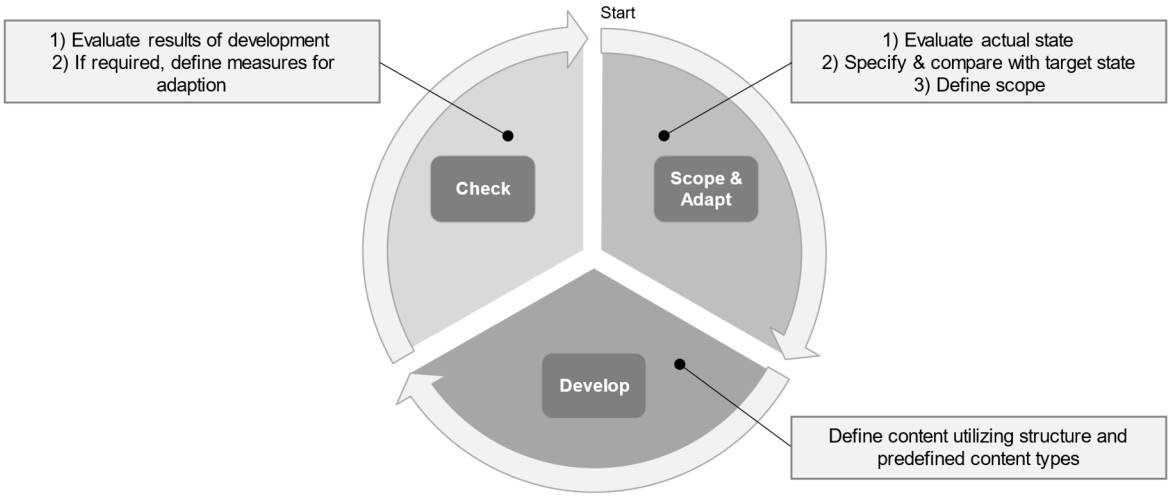


Figure 52: Iterative procedure for creating architecture content

Scope & adapt combines the definition of the scope for the definition of new content on the one hand as well as the adapted scope for improvement of already defined, but insufficient content on the other hand. In the scope & adapt step the actual state is evaluated first, second the actual state is specified and compared with the targeted state, and third, based on that evaluation, the scope for the development of content is defined. The target state shall also consider, which contents might be required in following development steps. The development step comprises the creation of content by the relevant stakeholders. After a set of content results is created, the third step is processed. Within the step check, the developed content is evaluated. The new current state is again evaluated against the target state specified within the first step. Based

on the evaluation it is decided by the processing stakeholder whether or not the created results are sufficient with respect to the specified target state. If so, the definition of new content for other parts of the system can be conducted. If the results are deemed insufficient, the cycle is rerun. In that case, the scope for the next run is adapted for the development. After results have been adapted, the output is checked again. The procedure is executed until the results meet the specified target state.

The application of this generally described process in combination with the three specified steps for the creation of content within a view, has been merged as a methodology for the creation of architectural content of a view utilized during the application of the architecture framework concept. The methodology is described in the following and is shown in Figure 53.

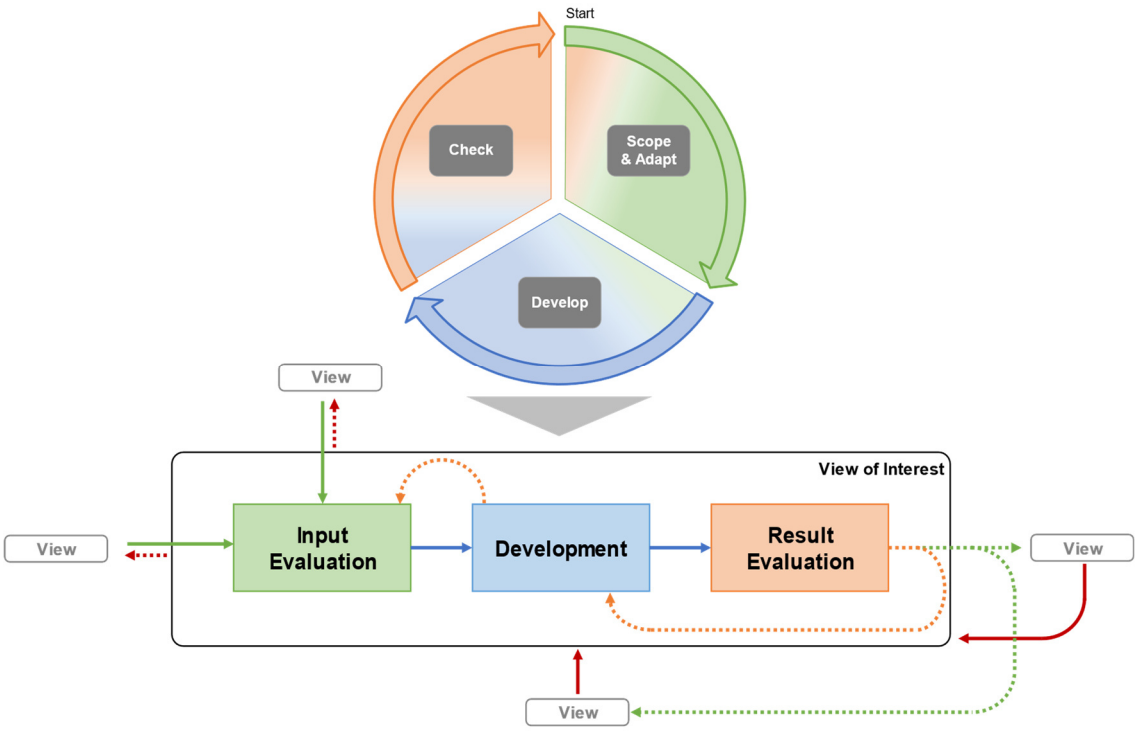


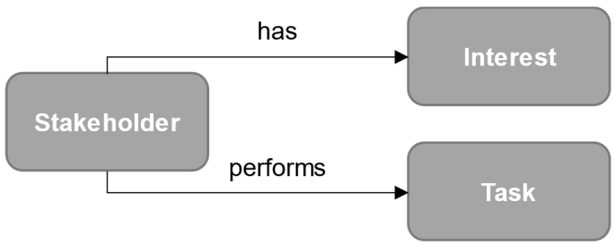
Figure 53: Architecture content creation method of core architecture framework concept

In the first step, the available input is collected and evaluated by the architect(s)/systems engineer(s). The input for the development emerges from the required views (as defined in section 5.4.3.3 and shown by green arrows within Figure 53). The decision whether the existing inputs are sufficient for the development process to be carried out can be difficult to judge objectively and will mainly be decided based on the experience of the processing architect/systems engineer(s). The decision that the available input is not sufficient can be made during the first step of evaluation or after the development process triggered by the processed check of

results. Such a decision might also result in the necessity that content within required upstream or higher-level views must be reworked (see red dotted arrows in Figure 53). This rework would then also go through the process steps scope & adapt, develop, and check. The effects how changes are considered within the architecture framework are described in section 5.4.3.5. If the inputs are sufficient or have been assessed as sufficient by comparing the actual state with the specified target state and the scope for development has been set, the creation of the architectural content of the view of interest can be started. As described before, the development process depends strongly on the individual considered view. Based on the predefined structures and content types introduced in 5.4.2.4., the concerned architect/systems engineer specifies the contents of the view of interest. As described in the view and content type section, the architect(s)/systems engineer(s) will utilize the provided element types, create instances of those types, and define relationships between the elements and if required between elements of other views.

Example – Content of Stakeholder Need View

If for example the domain and context view on GL1 have been defined and a horizontal shift from the domain to the requirement viewpoint is considered, within the requirement viewpoint the next to be defined view would be the stakeholder need view (compare Figure 40). Based on the input and consideration of the domain and context of the system of interest, the architect(s)/systems engineer(s) will look up in literature like [30] or define all relevant stakeholders in the stakeholder need view on GL1 based on experience. The architect(s)/systems engineer(s) shall utilize the contents predefined for the stakeholder need view within the architecture framework concept (as described in the annex Table 18 and Figure 95). The scope of content shall be defined beforehand. As a result, the architect(s)/systems engineer(s) might end up with results like shown simplified in Figure 54, representing a stakeholder, who has one or several interest, and performs one or several tasks. In the end all relevant stakeholders and their needs, represented in the form of their interests and tasks, shall be defined. In the same manner all other view content shall be developed when applying the architecture framework concept.



Example of potential content:

Stakeholder	Task	Interest
Customer (buyer)	Purchase specified system	Procurement on budget and on time
System operator	Proper application of system	Safe operation and the best possible execution
…	…	…

Figure 54: Simplified representation of stakeholder need view content

Afterwards, and in order to complete the development of the content of the view of interest, the results produced are checked. This involves checking whether the inputs from other views have been fully considered and whether all the necessary content is present in relation to the specified target state and scope of development. The results are then used as input for the definition of subsequent and subordinate views (see green dotted arrows in Figure 53). If an incompleteness of the results occurs during this check, the results are compared again with the target state for the development process and the initial input (orange dotted arrows). If the results are insufficient due to incomplete input, the upstream and superordinate views must be updated. If the input is sufficient and complete, the design process can be re-run and the results can be checked again. This loop can be repeated until the result is satisfactory to the current understanding of the reviewing stakeholder. Just like the stakeholder need view considered as an example and shown in Figure 53, subsequent views can also provide inputs or triggers for updating the content of the view of interest, if the content provided is insufficient (red arrows). These triggers can arise, for example, in the context of reuse or in case of changes in the life cycle of the system of interest. An associated process concerned with this situation is defined in the next section.

Concluding, it shall be mentioned, that as shown in color in Figure 53, the procedure and the individual steps cannot be clearly separated from each other, but merge smoothly into one another depending on point of view of the applying stakeholder. Nevertheless, when applying the architecture framework concept, care should be taken to ensure that all steps are followed in the sequence shown in order to obtain a complete and consistent architecture description in the end.

5.4.3.5 Consideration of Changes within Core Architecture Framework Concept

As briefly mentioned in the previous sections, situations may arise in which architectural content has to be adapted. On the one hand, this can be the case during the engineering of the system. On the other hand, the need for change may also arise due to other events in the subsequent life cycle phases of a system. This may be the case, for example, when errors occur during the operation of the system, when necessary spare parts are no longer available for older systems, when far-reaching adaptations of the system are necessary for a changed operational purpose, or when relevant innovations occur. Changes to a single system may also be relevant to a reference architecture, depending on the nature of the change and its relevance to other systems, and may also need to be considered there (for more information, see section 5.5.2.4). In order to carry out a consideration as comprehensive as possible of the necessary changes during application of the architectural framework concept, a supplementary method is introduced and explained in the following. The method is shown in Figure 55. The structure of the core architecture framework concept that has been already introduced, the introduced creation of content, and the pre-defined content element types remain unaffected and are partly utilized by this method. Via a repeatable process the method deals with identifying and adapting all affected views and contents within an architecture description.

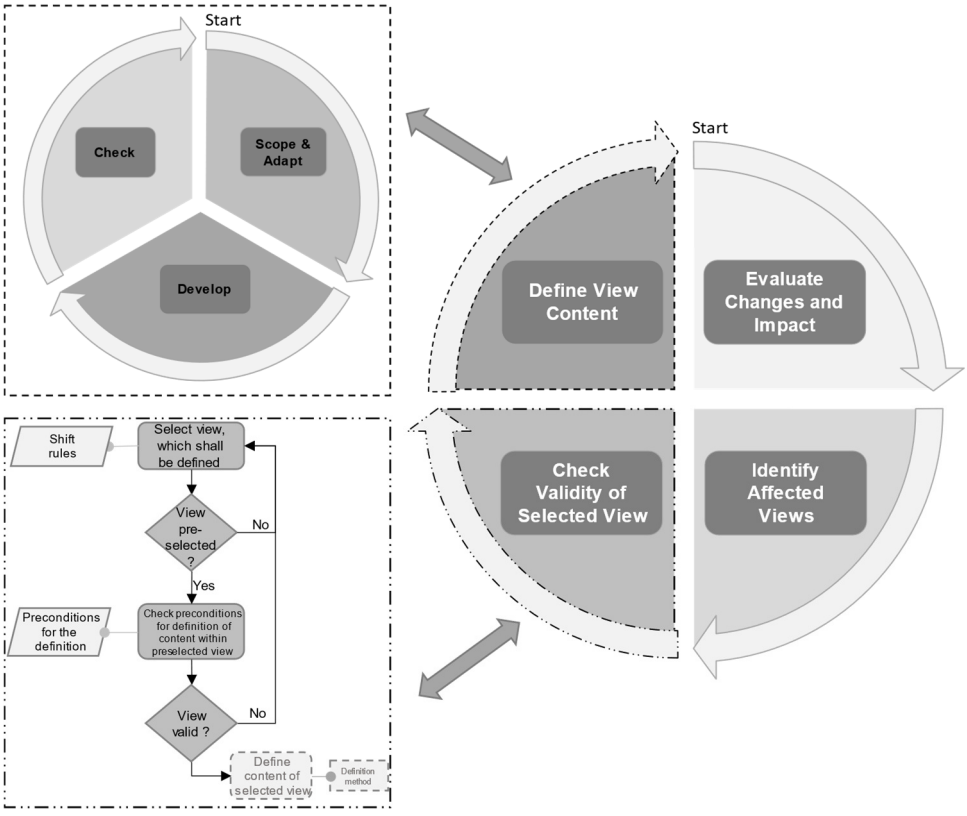


Figure 55: Change method of core architecture framework concept

The method consists of four repetitive steps. In the first step, the changes are evaluated and what effects they have on the views as well as on the contents of the architecture description. The precise examination of the effects of a necessary change is an important step in order to subsequently be able to adapt the affected views. However, the identification can only be supported to a limited extent by the core architecture framework concept and the creativity and experience of the processing stakeholder is needed. The stakeholder could, for example, be supported in the way that the core architecture framework concept is integrated into a model-based engineering landscape and thus, after relevant elements that have to be changed in the architecture description are identified, related elements/models are automatically highlighted and made available to the stakeholder. In this way, at least the possible influence of changes can be retraced reliably. However, tracing not only depends on the MBSE-idea itself but to a large extent on factors such as the quality of the models created and the linking of the individual contents.

In the second step, the relevant view(s) are identified whose contents must be changed. The individual views must then be processed in reverse order to the top-down approach. The sequence is reversed to comprehensively considered all views already defined (as shown in Figure 56). Therefore, the view with the highest granularity layer and the most advanced position in the viewpoint ranking is examined first. If changes are required, a valid view is selected for the third step. For both the first and the second step, a certain understanding of the stakeholder with respect to the system is required, because otherwise it is very difficult to classify the change(s) and their impact on the system description. In the third step, for the determination of the validity of a view the method introduced in section 5.4.3.3 is used. After the selection of the view to be changed has been made, in the fourth step, the content is changed/defined utilizing the architecture content creation method described in section 5.4.3.4 and Figure 53. After the fourth step is completed, the method starts again from the beginning, checking whether the changes made have any further impact on the architectural content of upstream and/or superordinate views. If the changes have further impact, the affected views are identified, selected, the updates are specified, and changes are made. This procedure is repeated until no more changes are required in upstream and/or superordinate views. Then, starting with the top granularity layer view in the earliest affected viewpoint, a top-down definition is performed again, to ensure consistency of results. If the architecture content is already defined, the content does not have to be completely redefined, but must be checked whether the changes still have undetected effects on the already specified contents. For this the described top-down approach as well as the method for the definition of

contents of the architecture framework concept are used. If the architecture has not yet been fully defined, the definition process can be proceeded once all intended content has been checked.

The change method as described above is shown in simplified form in Figure 56. The figure shows three states of an architecture description defined based on the architecture framework concept. The blue arrow describes the general direction of flow for a top-down definition. When a possible change occurs, the first possible view affected by the change is identified. Then, using the previously defined change method in the opposite direction to the flow direction, it is step by step traced back which views are affected by the change. This is shown by the red dotted arrows in picture ①. After a to be changed view has been identified and the content has been adapted, as described above, it shall always be checked whether an upstream or superordinate view is affected by the changes committed to the current view of interest. This is shown in picture ② by the orange dotted arrows. If no further views are affected, the described top-down definition process is started from the view at the top granularity level and of the viewpoint that occurs earliest in the structure of the core architecture framework concept, exemplarily shown in picture ③ by the green square and the green arrows. This starting point ensures that the changes made do not lead to inconsistencies or unknown errors in the architecture description.

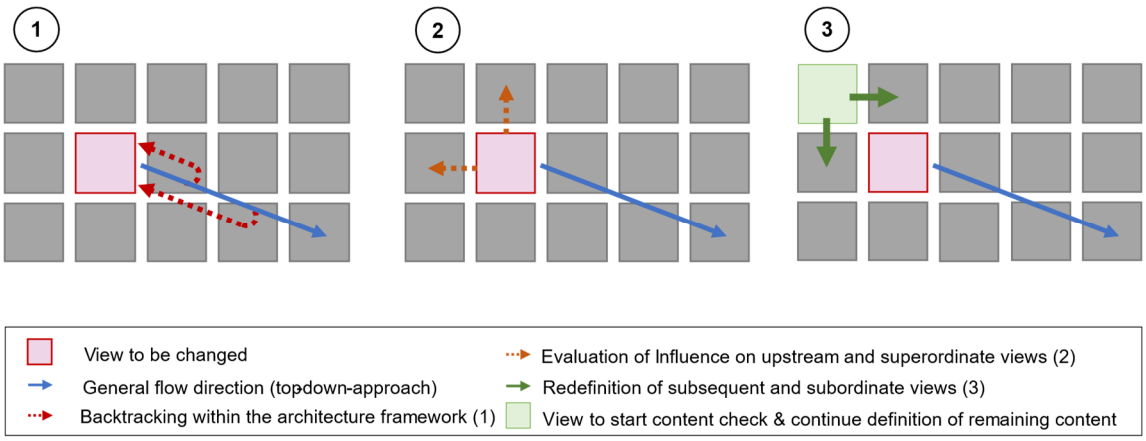


Figure 56: Simplified representation of consideration of changes to architecture content

The topic of changes and configuration management should be considered as an important part of (systems) architecting [61]. This means that, in addition to the consideration in the architecture framework concept, an accompanying configuration management should also be established, which is assumed as given in the context of this thesis and is not considered further.

5.4.4 Application of Core Architecture Framework Concept

To facilitate the applicability of the core architecture framework concept and the defined methodologies, a supporting guide for the definition of an architecture description and its modification is shown in Figure 57. The guide maps different states that are passed through during definition and adaptation. Based on this guide, the stakeholder(s) applying the architecture framework concept can work step by step through the definition process until the final architecture description is created. The guide first considers whether an architecture description should be created or an existing description should be adapted. If neither is the case, the application of the architecture framework concept should be stopped and the individual goals pursued should be re-evaluated.

If the goal is to create or modify an architecture description, the architecture framework concept can be applied as described in the previous sections. In the case of creating an architecture description from scratch, the stakeholder(s) should first consider the selection of a valid view as a starting point in the architecture framework concept. In order to do so the view validity determination method can be applied. The method is described in section 5.4.3.3 and in Figure 50. After a view has been selected to be defined and the preconditions for the definition of content of views have been met, the actual definition can be started/continued. For this purpose, the architecture content creation method is applied to the selected view of interest (section 5.4.3.4).

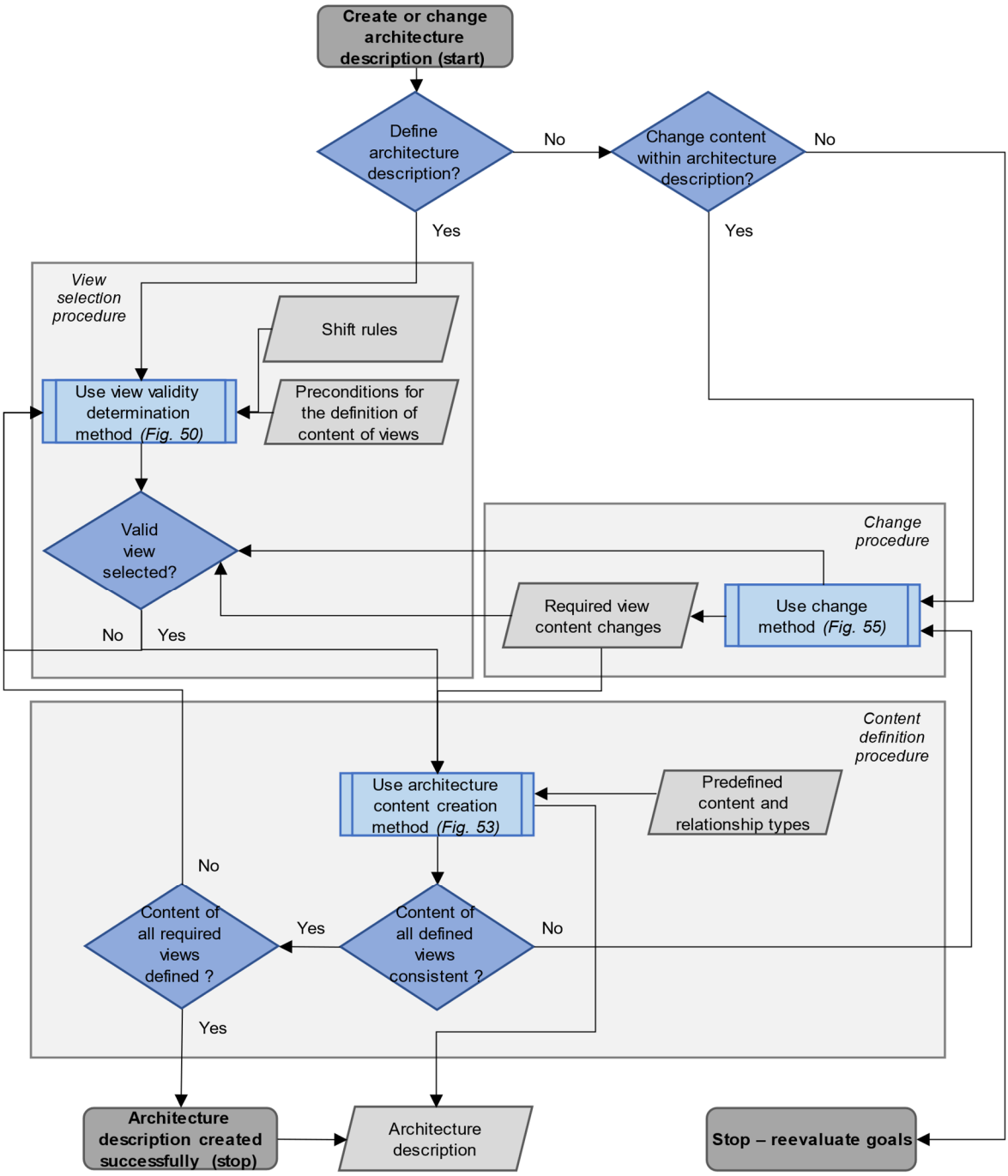


Figure 57: Application of core architecture framework concept

After the specific architecture description has been defined in the view of interest, it should be checked by the stakeholder whether all contents in the current view of interest and in all other views defined so far are consistent. The architecture framework concept does not explicitly consider this step, as this assessment is mainly based on the experience and skills of the executing stakeholder(s). If all contents are consistent and no errors occur, it must be assessed whether all contents of the architecture description have already been created. If yes, then the application of the

architecture framework concept can be stopped. If not all views and contents have been defined yet, the view validity determination method is applied again until another view of interest has been selected, which then also runs through the process described above. This loop is repeated until all views and the architecture description are defined.

However, if inconsistencies occur after using the architecture content creation method or if an adaptation of the architecture description is pursued from the beginning, the change method can be applied. This method is shown in section 5.4.3.5 and overlaps as described with the architecture content creation method and thus also with the view validity determination method. For the necessary adaptation, appropriate changes are then formulated and carried out with the help of the creation method. As shown in Figure 56, this loop is running until all view contents are consistent and error-free. After that, the completion of the architecture description can be continued as described above.

5.4.5 Conclusion on Architecture Framework Concept for Creation of Architecture Descriptions

In previous sections, a core architecture framework concept for creating architecture descriptions of production systems has been presented and described. The presented framework concept can be used for the creation of system and reference architectures alike. As described the architecture framework application differentiates between single system and a group of systems. As a result, the different inputs determine whether a system or reference architecture is specified. The applied procedure itself does not differ. In order to support the stakeholder applying the architecture framework concept with the task of defining architectural descriptions, the shown contents and a guide for application have been specified, which are shown in Figure 58. These most important sections and related results of the core architecture framework concept are shown as well.

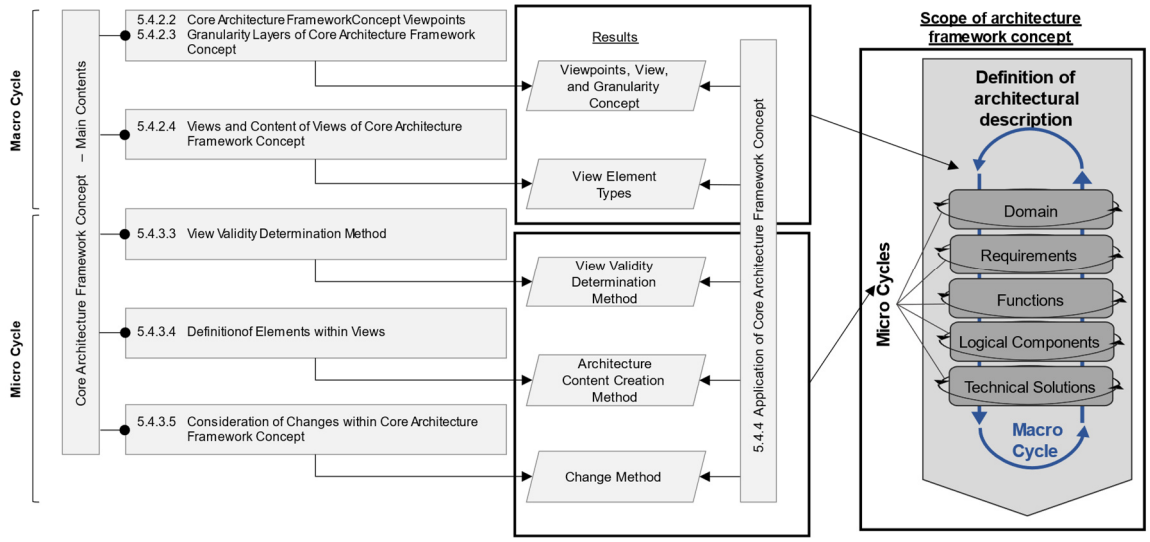


Figure 58: Overview of core architecture framework concept for the definition of architectural descriptions

Based on the requirements defined for the architecture framework and the analyzed existing frameworks, a basic structure for the framework was defined. This structure consists of five viewpoints as well as a granularity layer concept, which enables the specific consideration of individual views. The structure provides the basic relationships between the individual elements of the framework. Based on the structure, the content types to be considered within the individual views were defined. Due to the given structure, the architecture framework spans a two-dimensional problem and solution space between viewpoints and granularity layers. The structure and its viewpoints represent the general basic engineering procedure for creating an architecture description (macro cycle). After the structure and content types of the architecture framework concept were specified, the framework was supplemented by several methodological blocks (micro cycle) that consider the shift between views, the procedure for the definition of content within views, and the consideration of changes. The methods connect the structural and content types of the core architecture framework concept. The methods are all designed to be compatible with each other. In addition, the methods that need to be applied more than once are designed to work in a loop so that the process can be run through multiple times without having to apply additional steps, to keep the complexity level low. The defined conditions and rules for the view validity determination method reflect the selected top-down approach and contribute to the fact that all necessary views and relevant inputs are considered during the definition. The creation method and the change method build on each other. As the name already describes, these are used during the definition of content within a view and the consideration of necessary changes. All elements and methods are

considered in the specified guide for the application of the core architecture framework concept.

In conclusion, the core architecture framework concept provides a clear structure, regulates the relationship between the individual elements, and methodically enables the creation and adaption of relevant content. With this result, the specific requirements on the core architecture framework concept with regard to the research questions RQ1 and RQ2 were elaborated. It also forms the basis for the requested consideration of reference architecture descriptions for the definition of system architecture descriptions. This extension of the core architecture framework concept is described in the following sections.

5.5 Transition from Reference to System Architecture Description

After the components of the core architecture framework concept have been presented in the previous sections, this section presents a method for the transition between reference and system architecture descriptions as an add-on to the core architecture framework concept. This method is referred to as architecture framework transition method or in short transition method.

5.5.1 Derivation of Architectural Content – Actual State and Challenges

Based on the assumptions made in the previous sections, reference and system architecture descriptions of comparable structure can be created by individual application of the core architecture framework concept. Based on the fact that, the abstraction level was specified for both reference and system architecture before the definition, a two-dimensional design space defined by granularity layers and viewpoints results. The made basic statements are shown in Figure 59.

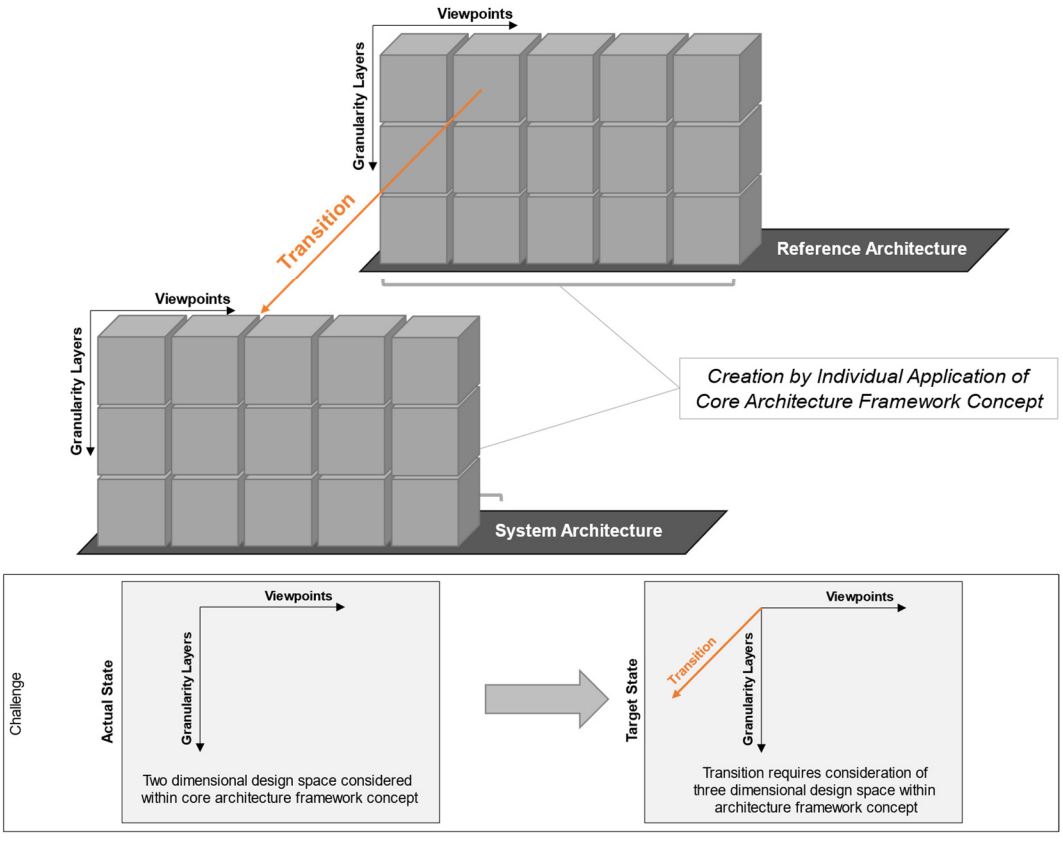


Figure 59: Initial situation of transition between reference and system architecture

Furthermore, it is considered that the use of reference architecture descriptions in connection with the application of the core architecture framework concept results in the additional dimension transition in the design space. This results in a shift from a

two-dimensional into a three-dimensional design space. This new design space is not covered by the current scope of the core architecture framework concept. The resulting main challenge is to take this additional dimension into account while at the same time ensuring the applicability of the defined core architecture framework concept.

5.5.1.1 Dependencies of required and available content

In addition to the challenge of transition, the dependencies between required and available content shall be considered. The different potential scenarios resulting from dependencies between required and available content have already been shortly teased in the section 0 and are considered in detail in the following.

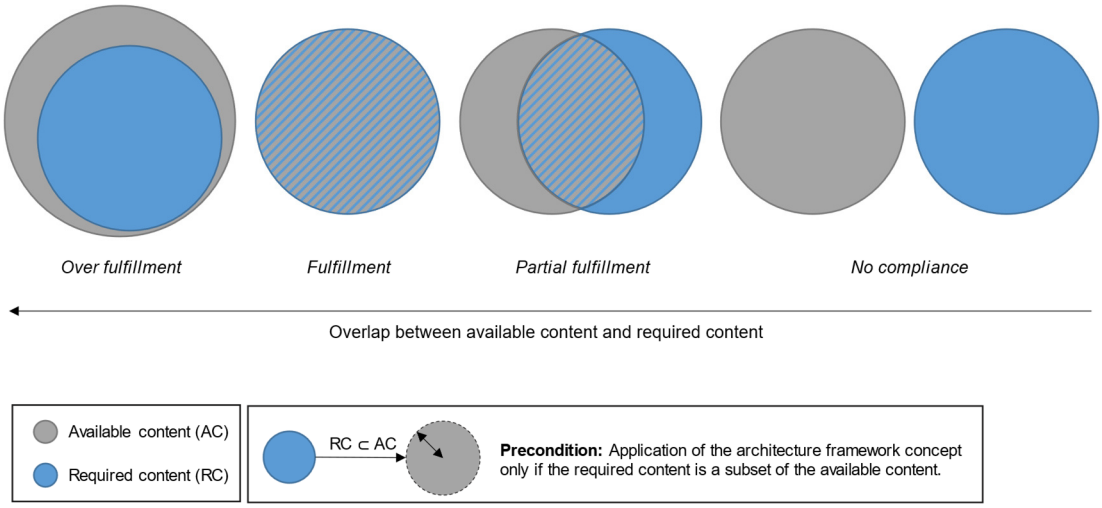


Figure 60: Dependencies of required and available content (adapted from [181–183] in [137])

Within this thesis, the available content expresses the architectural content of an architecture description provided by a reference architecture (shown as grey circle in Figure 60). The required content represents the expected and planned content of an architecture description for a considered system of interest (blue circle). As described in the assumptions in section 5.2, it is expected that when an architecture framework is used to consider reference architecture descriptions for deriving system architecture descriptions, an applicable reference architecture is selected and used. Thus, the required content for the system architecture is always a subset of the available content and the system of interest must be part of the group of systems considered within the reference architecture. It is important to note at this point that just because a system of interest is part of the system group considered in the reference architecture, the required content for the system of interest must not automatically be completely covered by the available content of the reference architecture. The content coverage

depends mainly on the concept pursued in the reference architecture and whether the reference architecture is more of an over-fulfilling "110 percent approach" or a "core approach" limited to the most relevant content. The resulting optimization problem of more content and supposedly better reusability compared to core content and therefore supposedly lower effort for the creation of the reference architecture is not pursued further in the context of this thesis but shall be considered in future research.

Based on Figure 60 and in opinion of the author, different scenarios can be considered in the context of transition and content utilization. These scenarios, or rather the resulting requirements, should be taken into account for the implementation of the transition method in the architecture framework concept.

Scenario 1: The available reference architecture description content corresponds completely with the required content of the system of interest or over fulfills it. Following the generally given assumption by [181–183] in [137] on reuse, the scenario described above would either require minimal changes or imply unchanged reuse of content. As the reference architecture always refers to a group of systems and in order to guarantee the validity of the architecture description for a larger group of systems, in most cases the available content has to be considered in a more abstract way than the necessary required content for an architecture of a specific system. Therefore, from the author's point of view, unchanged reuse might be possible, but an adaption of available content to the specific system of interest would be more likely. As previously described the focus of most reference architectures is on functional and logical considerations (including all required upstream contents), which means that, at least in these areas, the need to add content should be relatively small.

Scenario 2: The required content for the system of interest corresponds to a certain extent with the available reference architecture description content. The consequence, following [181–183] in [137], would be that some individual contents might be transferred, some might be adapted to the specific system of interest requirements, and some must be created from scratch. To achieve the completeness of the required content, the individual content components must be combined into a consistent architecture description.

Scenario 3: The required content of the system of interest does not correspond with the available reference architecture description content at all. This could be the reason if the system of interest is not part of the group of systems considered by the reference architecture. The consequence would be, that a

different reference architecture must be used for the design of a specific system architecture description or, if not available, that the system architecture must be defined from scratch completely. In the latter case, the core architecture framework concept can be utilized. Since a definition is possible through the application of the core architecture framework concept and the assumption has already been made that a matching reference architecture is always applied, the scenario that there is no match between required content and available content is not considered further in detail in the following.

In summary, since scenario three was excluded for the reasons described above, the consequence for the extension of the core architecture framework concept is to consider a mixture of scenario one and scenario two. This means, as already indicated in section 0, that available and required contents must be compared in order to assess which scenario is involved. This determines the starting point for all further tasks during the transition. Depending on how the contents overlap, the pure transfer of contents, the adaptation and supplementation of existing contents, and the definition of new contents that are missing in the reference architecture description must then be considered in the transition extension for the core architecture framework concept.

5.5.2 Architecture Framework Transition Method for Core Architecture Framework Concept

In the following, the architecture framework transition method for the consideration of reference architecture description content during the application of the architecture framework concept for the definition of a specific system architecture description will be in short referred to as transition method. Made assumptions and design decisions for the method are introduced in section 5.5.2.1., followed by the documentation of the actual transition method in section 5.5.2.2.. After the completion of the transition method its application in combination with the core architecture framework is exemplarily shown in section 5.5.2.3. In addition to the method and in comparison to the core architecture framework concept the topic of change is examined in section 5.5.2.4. A guide for the application of the architecture framework concept is given in section 5.5.3.

5.5.2.1 Made Assumptions for the Architecture Framework Transition Method

Most of the assumptions have already been defined in section 5.2 and connected to the creation of the core architecture framework concept in the previous sections. Additional assumptions with respect to the transition method are summarized in Table 7 and have been made, as in section 5.2, based on the discussion with experts of the

considered scientific field, the relevant ISO guidelines and standards, such as VDI 3695, the contents and results of the research project CrESt and the presented state of the art. These assumptions are intended to restrict the solution and to create consistent and reproducible conditions for the application of the defined method.

Table 7: Additional assumptions for the architecture framework transition method

	Additional Assumptions
Architecture Framework	• A suitable reference architecture with a relevant intersection with the required content for the definition of a system architecture description exists and is used. Predefined reference architecture description content is used for as many viewpoints and views as possible. • In case that the technical reference architecture is not available, the technical architecture description of the system of interest is defined from scratch. • The components of the system and reference architecture description are structurally and content type wise comparable, as they both focus on similar domains. • The problem space of the system of interest is described individually before the transition, to ensure that the problem space considerations are not influenced by predefined contents. For this purpose, the core architecture framework concept shall be used. After the individual definition of the problem space of the system of interest a comparison with the reference architecture description problem space should be carried out in order to assess the completeness of definition of the system of interest problem space. • The transition method is defined in a way that the structure and methodologies of the core architecture framework concept are utilized. • For the transition available and required contents have to be compared, available reference architecture contents potentially have to be adapted for application, and missing required content must be defined from scratch. For the description of a single view of a system of interest, the different available contents, additions, and from scratch definitions have to be summarized. In order to take these transition steps into account and in order to use the core architecture framework components, the individual steps are considered in a separate manner and linked to each other at relevant points.

The main assumptions for the transition made within the table above are depicted in Figure 61, which shows the predefined problem space of the system of interest, the input for the transition, the same structure of reference and system architecture (due to the applied architecture framework), the transition between the refence and system

architecture, as well as the potential abstinence of a technical reference architecture description.

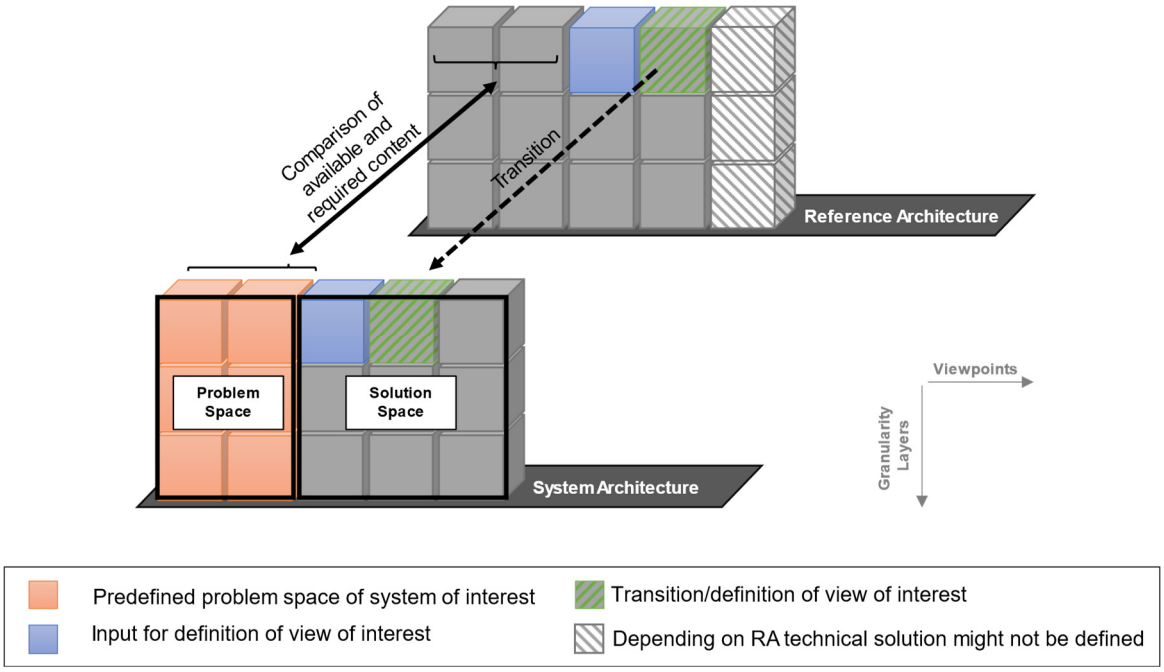


Figure 61: Main assumptions with respect to the architecture framework transition method

5.5.2.2 Architecture Framework Transition Method

Based on the sum of assumptions, design decisions, and predefined core architecture framework content, the transition method shown in Figure 62 was developed for the consideration of reference architecture description content as a basis for the definition of a system architecture description.

The architecture framework transition method consists of a three-layer structure (“SyA Creation Layer”, “Decision Layer”, and “RA Customization Layer”) and considers therein three main process steps (“(1) Evaluation of RA & SyA Inputs”, “(2) RA Customization”, and “(3) SyA Design”). The different layers of the method are intended to clearly structure the main steps of the procedure and simplify its application. As described within the design decisions and the previous sections, a system architecture description is created based on available reference architecture description content, which might be adapted to fit the system architecture. The evaluation of required and available architecture description content, statements about the necessity for reuse and the adaption of available content, and/or the definition of required nonexistent content are made in the “Decision Layer (B)”. All considerations related to RA are allocated to the “Reference Architecture Customization Layer (C)”. In addition,

nonexistent content might be supplemented and all contents for the definition of a view of interest have to be combined into one solution. All such considerations are made within the “System Architecture Creation Layer (A)”.

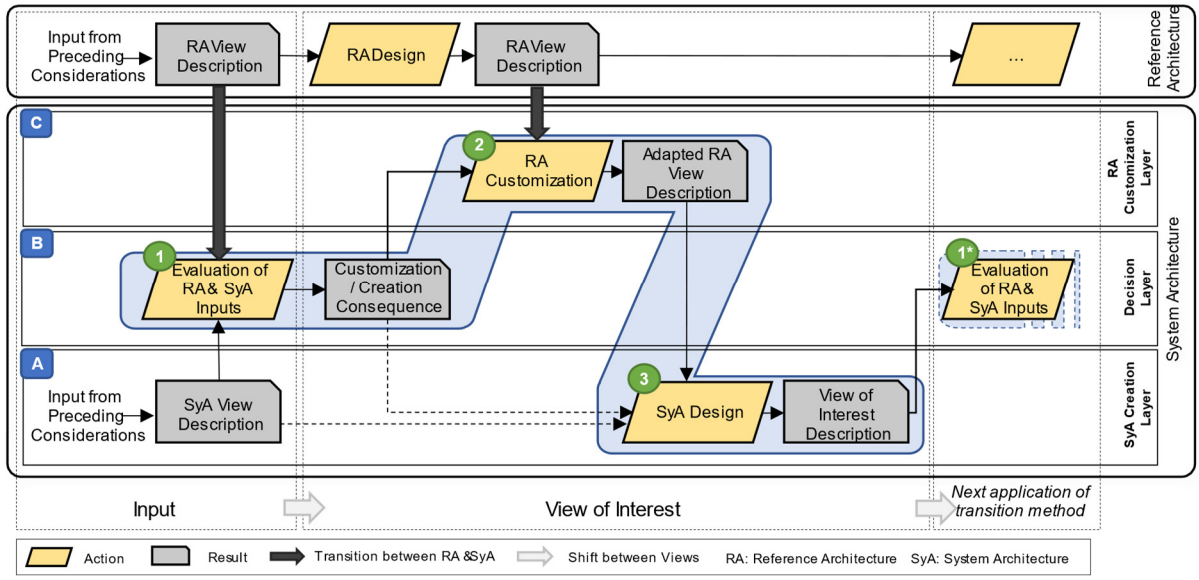


Figure 62: Architecture framework transition method - adapted from [147, 148]

The three main steps of the method are distributed over these three layers, which result mainly from the general consideration of the relationships between available content and required content (as elaborated in section 5.5.1.1). Step (1) "Evaluation of RA & SyA Inputs" represents the starting point of the transition and compares potentially relevant inputs of the reference architecture description with the inputs for the definition of the view of interest. As a result of this analysis, "Customization/Creation Consequences" can be derived. Thus, the evaluation result provides information about which parts of the reference architecture description can be reused and, if necessary, have to be adapted (customization consequence) and which contents for the definition of the view of interest have to be defined from scratch (creation consequence). Based on this evaluation, the existing reference architecture description contents are first copied and, as needed, adapted to the specific view of interest. In addition, the relationships to other relevant elements within the system of interest are established. This is done in step (2) "RA Customization". The result is a reference architecture view description adapted to the view of interest ("Adapted RA View Description"). Depending on the customization/creation consequence, this partial result must be supplemented for the view of interest. If all required contents are available, the result from step 2 also represents the overall result for the view of interest. In this case step three could be omitted. If this is not the case and specific

additions are required, the "View of Interest Description" is defined in step (3) "SyA Design". This is done based on the adapted reference architecture view description and taking into account both the defined customization/creation consequence and the input of the relevant system views (“SyA view description”). In this step, the missing content is added while the additional content is related to the existing content. The newly created description of the view of interest then represents the new input for the definition of the next view of interest. The next application of the transition method is exemplarily shown in the form of step (1*) in Figure 62.

The allocation of the above-described procedure is exemplified for a transition between a functional reference architecture view description and a functional system architecture view description in Figure 63. The transition method takes place on the second granularity layer, which links the view of interest (blue box) with the inputs from views within reference and system architecture (orange box). As described, in a first step, the inputs of the reference architecture and the system architecture are compared and evaluated, to determine the content of the reference architecture which can be reused or has to be adapted and content which is potentially missing, which has to be defined from scratch. Based on the connection between the elements of the views within the requirement viewpoints and the elements within the functional view of the reference architecture, in the second step, based on the customization consequence, the functional view of the reference architecture (light blue box) can be customized to be applicable within the system architecture description (blue box). In the third step and with respect to the inputs of the system of interest, the determined missing content is added to the result of step two. The sum of all contents forms the result of the third step, which is in this example the functional view of the system of interest (dark blue box).

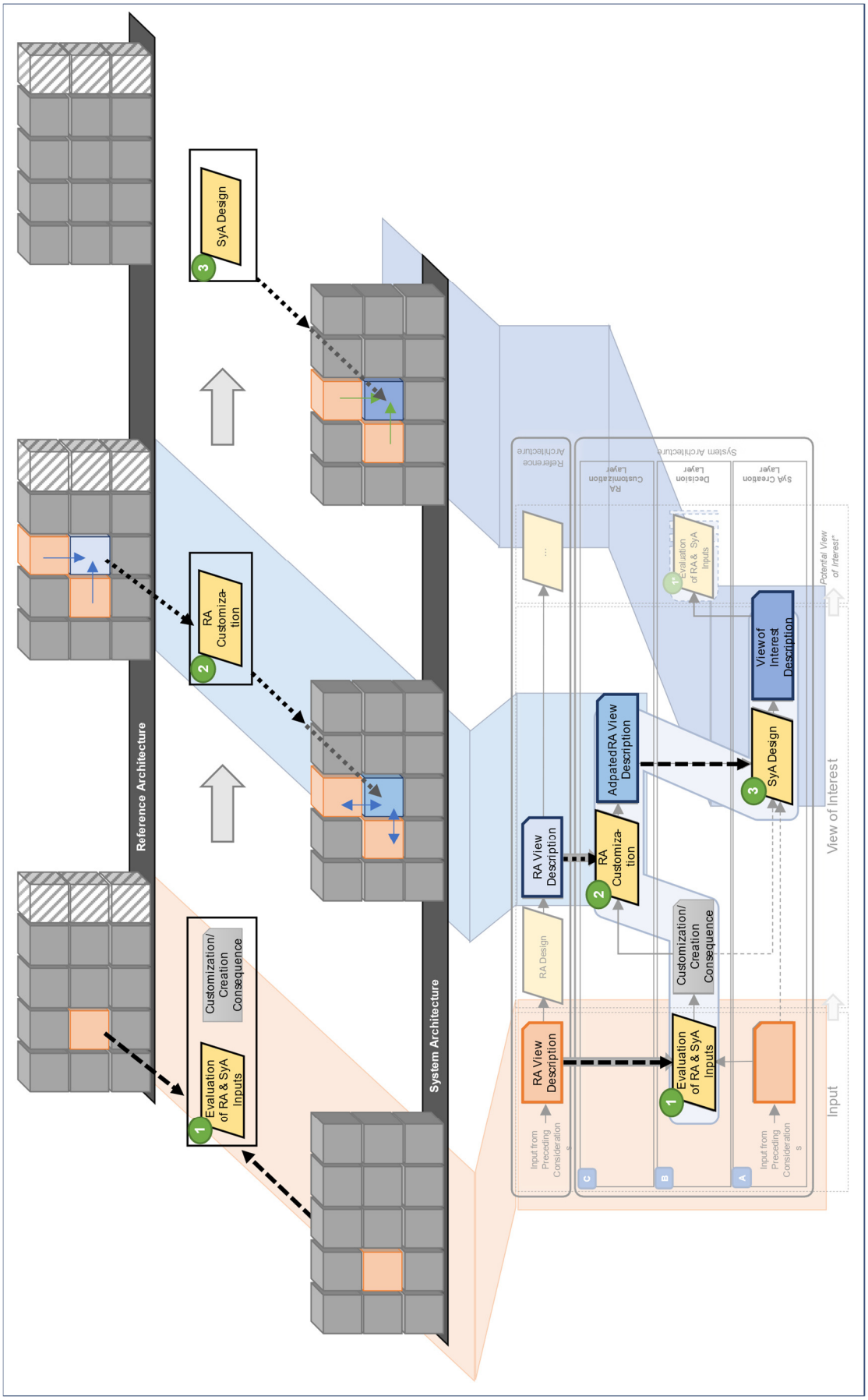


Figure 63: Exemplified allocation of architecture framework transition method to view

Due to the iterative design, the transition method can then be gradually applied to the definition of all views of the system of interest. In the problem space, however, the methodology is only applied after the separate and complete definition of the views of the domain and requirement viewpoint by using the core architecture framework concept. In this context, this partially represents a supplementary approach. If the technical viewpoint of the reference architecture is not available, the content is defined in the same way as the content of the problem space. The application is shown in simplified form in Figure 64.

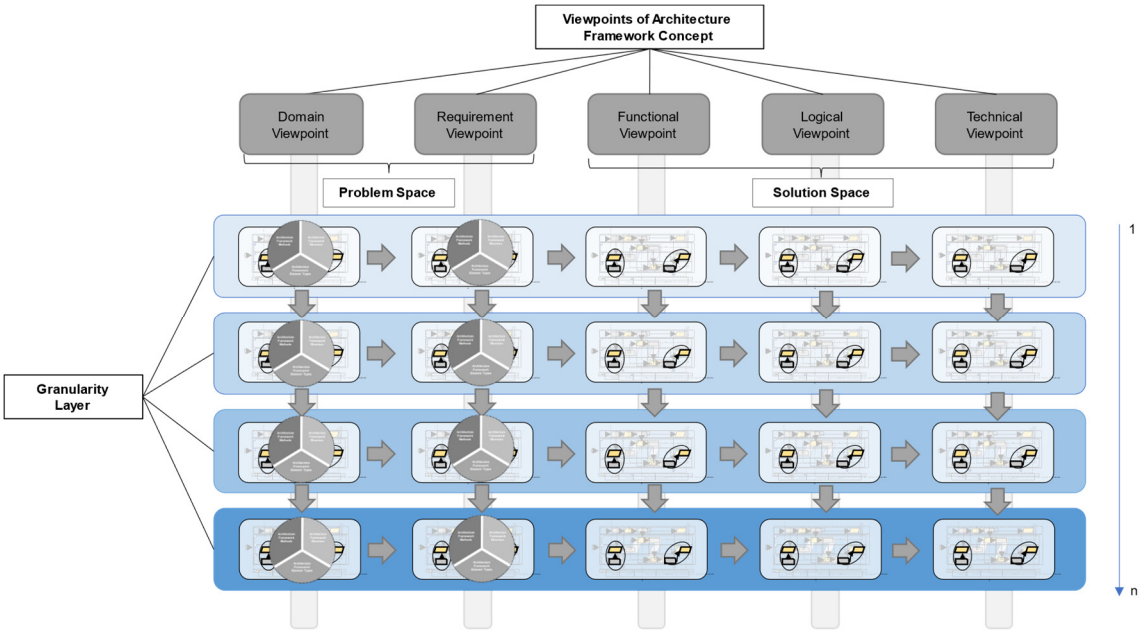


Figure 64: Example of the application of transition method for the vertical and horizontal definition of views within system architecture description

In Figure 64 it is assumed that the transition shown in Figure 63 is covered by the represented transition method. The overlap between granularity layer and viewpoints is deliberately shown, to emphasize the point, that the transition method applies to all the different views within the architecture framework concept. Of course, as previously defined, for the actually performed definition only one view is considered at a time.

5.5.2.3 Example of Utilization of Core Architecture Framework Concept Components within Architecture Framework Transition Method

The overall context of all defined components of the architecture framework concept is presented in section 5.5.3 in the application guide. Nevertheless, in order to clarify the application of the transition method better, the creation of content in combination with methods from the core architecture framework concept, in particular the architecture content creation method, is shown in Figure 65. At the beginning of the

transition, comparable to the core architecture framework concept and as shown in Figure 57, a view of interest must be selected. The same procedure as described in section 5.4.3.3 applies for the selection of a suitable view. The application of the view validity determination method without adaptions is possible since the input of the reference architecture can be assumed as given and therefore is not considered as relevant precondition for the selection. The view of interest is selected, taking into account the possible shift directions between views, and the preconditions used to check whether the inputs required for the definition of the view of interest have already been defined in the system of interest. Analogous to the core architecture framework concept, this procedure must be performed until a valid view of interest is found. If a view of interest has been defined, the first step of the transition is to define the customization/creation consequence (1), the second step is to define the adapted reference architecture view description (2), and the last step is to add missing content and integrate all contents into one view of interest description solution (3). As all three steps consider inputs, their processing, and provide a corresponding output, the idea that the creation method defined in the core architecture framework concept could be used for this definition arose due to its similar structure. The difference in application compared to the core architecture framework concept, as shown in Figure 57, is that the architecture content creation method is always applied iteratively, but not just once per view, but for each of the three transition method steps. For the first step of the transition method, the “evaluation of reference and system architecture inputs”, the input is evaluated and a corresponding scope for the development is defined. However, in connection with the first step of the transition method, the development does not describe the formulation of an architecture description based on predefined element types, but the definition of the customization/creation consequence. It is thus specified in the “scoping” step of the creation method and under consideration of the existing reference and system architecture descriptions, which has to be considered in the context of the customization/creation consequence (target state). The definition of the customization/creation consequence takes place during the “development” step of the creation method. The evaluation of the result by comparing the achieved actual state with the specified target state is then performed in the step “check”. If the specified content does not correspond to the desired target state, the architecture content creation method is repeated iteratively until the desired state is achieved.

Regarding the creation of the customization/creation consequence, it shall be mentioned that no separate process has been defined that takes into account the comparison and the necessary unambiguous specification of architecture elements. Those considerations represent an independent research area, which would exceed

the thematic scope of this thesis. It is therefore assumed that the comparison will be performed by the applying stakeholder, who will then produce an appropriately complete list/table or similar, identifying existing content that may need to be adapted, as well as new content to be created. However, the comparison is aided by the fact that in the previous sections it was decided as a facilitating assumption that reference architecture and system architecture use the same structure and element types as introduced by the architecture framework concept. This can at least ensure that only the same element types are compared. The extent to which the resulting instances match is then judged by the respective stakeholder.

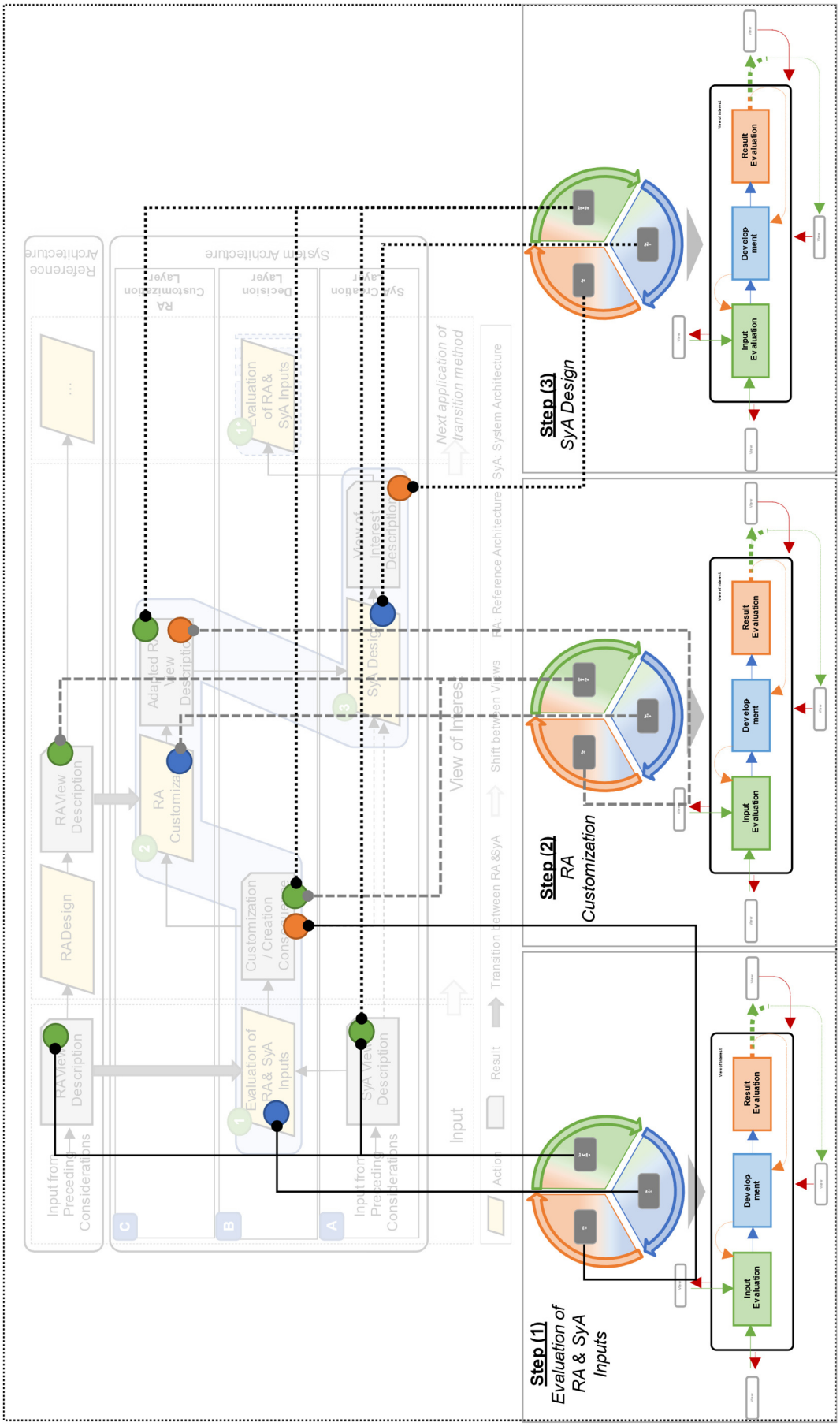


Figure 65: Example for allocation of core architecture framework concept to transition method

For the second and third step of the transition method, the architecture content creation method is applied in the same way as described in section 5.4.3.4. In comparison with step one of the transition, both steps two and three are concerned with the adaptation and creation of architecture descriptions, utilizing the predefined view element types, and therefore better fit the architecture content creation method. For the second transition step, the reference architecture customization, the input for the creation method is not only composed of the existing view contents of the reference architecture, but also of the customization/creation consequence defined in the first step. This must be included as additional input so that the stakeholder performing the development step can carry out the customization of the reference architecture with the correct scope. For the reference architecture customization, a copy and modify approach is followed, in which the corresponding reference architecture content of a considered view is copied and adapted accordingly. Adaptation can be, for example, created by deleting unnecessary content. It should be noted at this point that although the basic content and relationships are defined based on the specified element types of a view, the assessment of the individual instances and whether they are relevant in the context of the customization/creation consequence depends on the decision of the executing stakeholder. The subsequent customization and instantiation are also subject to the stakeholder, who is thus mainly responsible for the final shape of the view of interest. Concluding the result of the creation process, the adapted reference architecture view description, is checked and, if necessary improved.

Step three of the transition, the system architecture design, is carried out in the same fashion as step two. The main difference is that not only the customization/creation consequence is used as input, but that also the required input from relevant system views and the adapted reference architecture view description created in the second transition step are considered. Based on the results of the second step and the other inputs, the existing content is linked with the still missing content of the view of interest in the development step of the creation method. Created content is then checked again as described above and, if necessary, corrected by iteration. After the comprehensive definition of the description of the view of interest, the next view can be selected as described at the beginning of this section and the transition method in combination with the architecture content creation method can be applied in the same way.

In general, it should be noted at this point that for comparison between reference and system architecture, it might be necessary to perform a corresponding back-tracing over several views, to identify potentially reusable content.

5.5.2.4 Consideration of Changes within Architecture Framework Concept

When considering the resulting changes, and assuming that the framework uses reference architecture content and defines system architecture content alike, a distinction must be made whether changes to the reference architecture and the system architecture occur before, after, or during the transition and the creation of a system architecture description. For a better distinction three possible scenarios are shown in Figure 66.

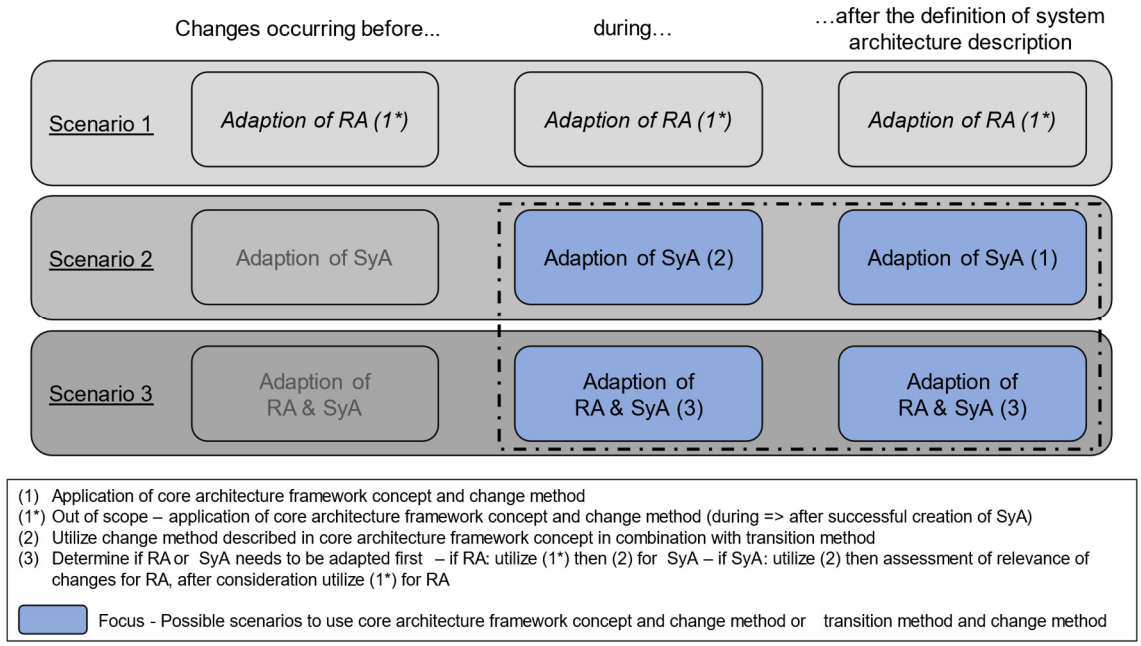


Figure 66: Potential change scenarios

The first scenario describes a case where changes become necessary only with respect to the reference architecture description, in scenario two with respect to the system architecture description, and in scenario three with respect to both. Since the focus is on the creation of the system architecture descriptions and only on the use of the reference architecture description, a few limiting assumptions are made. In principle, it is assumed that before the system architecture description is defined, no system architecture description exists that would need to be adapted.

Before: Therefore, scenarios two and three are omitted at the time before the definition. Furthermore, the assumption has already been made that a suitable reference architecture description is always available and applied. Therefore, scenario one prior to the definition of the system architecture description is actually out of scope. Nevertheless, the reference architecture description can be adapted by application of core architecture framework concept and change method (1*).

During: Due to the assumptions made, scenario one is also out of scope during the definition of the system architecture description. Should the need for change still arise after the system architecture description has been successfully defined, the reference architecture description can again be adapted by the application of core architecture framework concept and change method (1*). If only changes to the system architecture description itself are necessary during the definition (scenario two), the change method described in the core architecture framework concept in combination with the introduced transition method can be utilized (2). If scenario three occurs during definition, which is a borderline case in terms of assumptions, the architect must first assess, based on the specific situation, whether the changes should be made first to the system architecture description or to the reference architecture description (3). If the reference architecture description shall be changed first, the procedure described for (1*) shall be utilized and then for system architecture description (2) can be utilized again. If the system architecture description shall be changed, first utilize (2), then an individual assessment is needed on which changes can be abstracted to be considered within the reference architecture description. After assessment the reference architecture description can be adapted utilizing (1*).

After: For the time after the definition of the system architecture description, the same procedure considered before and during creation of system architecture description applies for scenario one. If the adaptation of a system architecture description is required after its definition, the core architecture framework concept and change method can be applied (1). For scenario three, the same procedure applies after definition as during definition (3).

5.5.3 Application of Architecture Framework Concept

As already introduced for the core architecture framework concept, a corresponding guide is also formulated for the application of the extension, the transition. This guide is shown in Figure 67 and pursues the same goals and uses partly the same contents as the guide in Figure 57 of the core architecture framework concept. Basically, in a first step, a distinction is made whether an architecture description is to be created using a reference architecture description or not. If this is not the case, a further distinction is made whether an architecture description should generally be created or an adaption of already defined contents shall be made. Depending upon the decision in the case of the general consideration of architectural description, reference is made to the core architecture framework concept and to the change method in terms of

adaptions. If neither of these topics are considered, the definition efforts should be discontinued, and the goals set should be re-evaluated and adjusted if necessary. However, if an architecture description is to be created using a reference architecture description, the definition can be started by using the guide shown in Figure 67.

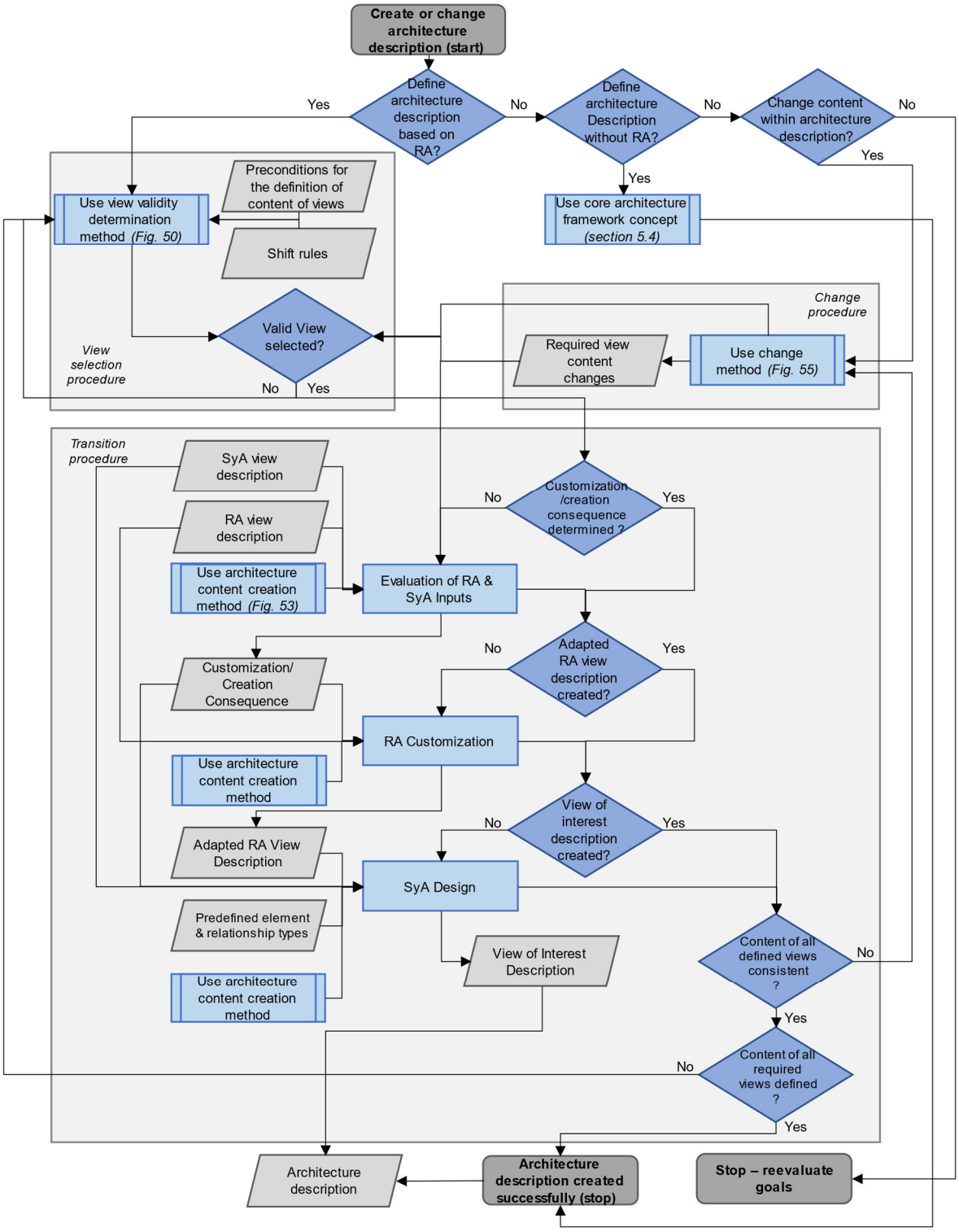


Figure 67: Application of architecture framework concept for definition of architecture description based on reference architecture

As within the core architecture framework concept, first the view validity determination method presented therein is used to check whether a valid view has been selected as the basis for the definition (view selection procedure). If a valid view has already been selected, the definition of an architectural description based on a reference architecture description can be pursued (transition procedure). If no valid view has been selected yet, the method can be applied again until a suitable view has been identified.

Once the view of interest has been selected, the transition can be started by applying the architecture content creation method to the individual steps of the transition methodology as shown in detail in Figure 65. First it is checked whether a customization/creation consequence has already been defined. If so, the definition can be directly continued with the second step of the transition. If this is not the case, the customization/creation consequence is defined using the described methodology and inputs. After the creation, step two of the transition is considered if an adapted reference architecture view description is not already available. If an adapted reference architecture view description has already been defined, the step can be omitted, and the definition process can be continued in step three of the transition. If the adapted reference architecture view has not yet been described, the methodology of the core architecture framework concept and the associated inputs are used again for the definition of the required content (“RA Customization”). Finally, in the third transition step, the view of interest description is defined using the same procedure described in the previous two transition steps. The upcoming tasks like checking completeness of views, taking changes into account, and completing the architecture description follow the same procedure as described in the guide for the core architecture framework concept (see section 5.4.4).

5.5.4 Conclusion on Overall Architecture Framework Concept

In summary, it should be noted that in this chapter a proposal for an architecture framework concept was made, which can be applied for the definition of system architecture descriptions based on the use of a reference architecture description. For this purpose, a core architecture framework concept was defined (see section 5.4), which describes structurally and methodically the creation of architecture descriptions in general. Building on this basis, the transition between reference and system architectures was then considered in this section. The aim was to ensure that the same basic methodological and structural approach could be used for different scenarios during the application of the framework. In addition, and to facilitate the application as well as to express the domain of interest, appropriate content element

types and structuring were predefined, on which the applying stakeholder can base the formulation of the specific architecture description. In order to test the validity of the architecture framework concept, the framework is prototypically implemented in a modeling tool in chapter 6 and evaluated in chapter 7 by applying it to an example.

6 Tool Implementation of Architecture Framework Concept

In this chapter, the prototypical implementation of the architecture framework concept is presented in the form of a domain-specific language within the modeling tool MagicDraw. The main relationships between the tool, modeling language, and methodology/framework are defined in section 6.1. and are shown in Figure 69. In section 6.2 the basic procedure and inputs during the development and implementation of the architecture framework concept in the form of a DSL (domain-specific language) within MagicDraw are described. Therefore, within section 6.3, the tool MagicDraw used for the implementation is introduced and briefly described. Section 6.4 specifies which components of the framework have been implemented prototypically and which components have not (yet) been considered. Concluding, section 6.5 presents some examples of the implemented DSL. The goal of this chapter is to illustrate the domain-specific language and the tool MagicDraw used for the implementation of parts of the architecture framework concept. The purpose is to create an understanding of the way of implementation and to pilot the use of the architecture framework concept in a realistic working environment. The implementation of a tool is the basis for the evaluation of the architecture framework concept and the proof of concept carried out in the next chapter. The content as well as the connection to previous and subsequent chapters is shown in Figure 68.

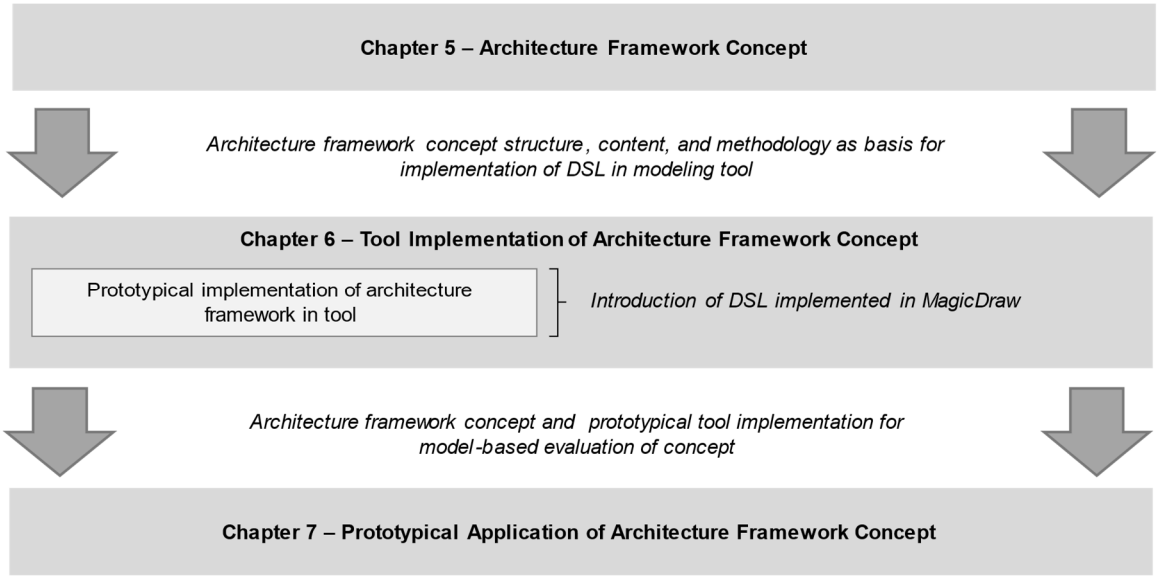


Figure 68: Overview of the contents of chapter 6

6.1 MBSE Tools and Languages

In order to meet the engineering challenges described in section 2.3 better, engineering tools are increasingly being used to provide support during development of systems [7]. Due to that development in SE/MBSE and especially in the areas of modeling language and tools as well as available and established architecture frameworks, the support of stakeholders regarding the definition of e.g., requirements or system structures has never been greater [15]. Therefore, the topic of modeling tools and modeling languages used for the creation of the required models are described below. The relations between relevant terms are shown in a simplified manner in Figure 69.

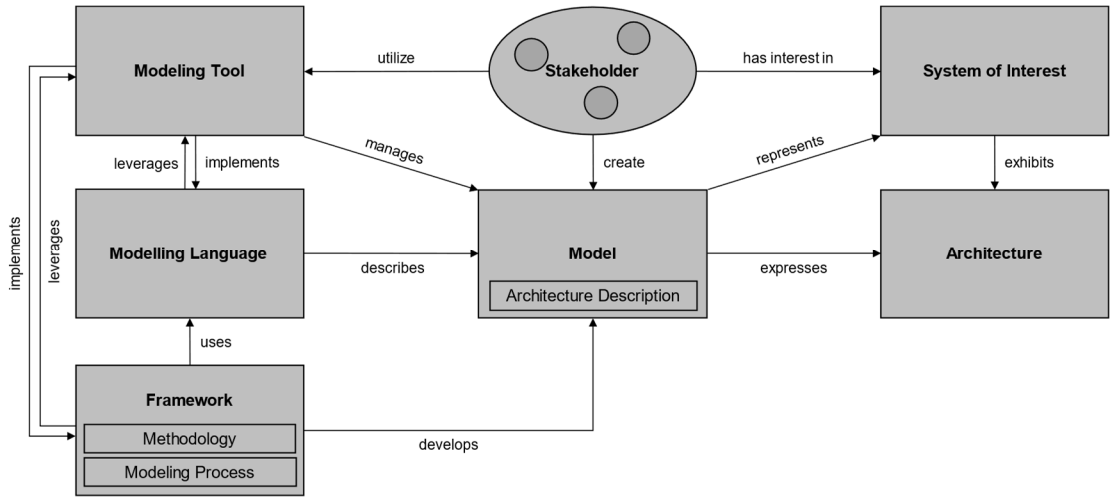


Figure 69: Tool and Modelling Language Environment (based on [30, 109, 184]

In the context of system development, the relevant stakeholders have different interests in relation to the system of interest and with respect to this thesis mainly towards architectures. The architecture describes the structure and interrelationships of the system (see definition in chapter 3). Considering model-based development, the system and specific aspects of the system are represented by one main model and/or several sub-models. The model also represents an architectural description of the system, which expresses the architecture. The required models are created by the stakeholders. In most cases, a modeling tool and the modeling language implemented in it are used for this purpose. A tool can be generally defined as follows.

Tool

A “[tool] (aids) assist the representation and/or documentation of knowledge” [154].

The model to be created is described using the modeling language which is managed within the modeling tool. The task of modeling languages is to provide means to represent the human perception of reality in a model and to provide a common starting point for all stakeholders involved [185]. A modeling language can be defined as follows

Modeling Language

A modeling language is defined as “[...] a textual or graphical language used to implement one or more related types of models” [38]. “Modeling languages are generally intended to be both human interpretable and computer interpretable and are specified in terms of both syntax and semantics. The abstract syntax specifies the model constructs and the rules for constructing the model from its constructs. [...] The semantics of a language define the meaning of the constructs" [26].

As not every tool can implement every possible language and depending on the application scenario as well as the modeling tool used, different modeling languages are available for describing the models. On the one hand, an architecture framework represents a modeling process, which considers what is to be done [184], on the other hand, theses frameworks serve as a methodological approach that describes how tasks should be handled [184]. The architecture framework is implemented in and leveraged by the modeling tool. The framework uses thereby, like the tool, the implemented modeling language and supports the development of the model as well as the architecture description contained in it. The use of such a tool and the framework it contains enables supporting the stakeholders in the context of their engineering tasks, which is one of the main MBSE goals.

By choosing a modeling language, it is determined which description means are available and which contents can be illustrated in a model [185]. Accordingly, the usability of the model is limited by the choice of the language. Depending on the application, one language is usually more suitable than another [185]. In general, modeling languages can be distinguished by different criteria, for example, textual or graphical languages, or by their scope of application. All the different modeling languages have specific advantages and disadvantages due to their characteristics and dependent on the application scenario they shall be applied on. Among the modeling languages and with respect to MBSE, a very rough distinction can be made between General Purpose Languages (GPLs) and Domain-Specific Languages (DSLs). Both are shortly described below.

General Purpose Language (GPL)

A general purpose language is a cross-domain modeling language with language constructs not restricted to one particular domain [186]. The language represents reality as well as relevant interrelationships and is utilized for implementing them in models.

Two commonly used GPLs related to systems engineering are the UML- Unified Modeling Language and the SysML - Systems Modeling Language [15], which is probably the most used modeling language in MBSE [15, 112]. In addition to SysML, several much simpler languages for modeling have been created, but have not gained extensive acceptance among the systems modeling community due to their limited utility as well as lack of precision and adaptability compared to established languages [15, 112]. Detailed information about SysML can be found among others in the following sources [110] and [187] as well as for UML in [74] and [188].

In addition to the MBSE benefits and implementation challenges described in section 2.3.5.1, the use of GPLs such as SysML and the associated tools can further hamper successful implementation. Obstacles arise by combining a user who is usually not a modeling expert with trying to apply a modeling language that is insufficiently methodologically supported and implemented in a tool designed for modeling experts. This leads to challenges for users and might results in a loss of productivity. In the worst case, users avoid the modeling language as well as the related tool and return to the known familiar and previously established document-centric approach. Due to those circumstances, some tool vendors have provided ways to adapt modeling languages to the needs of the users and the domains considered. [115]

The languages resulting from customization of GPLs, mostly to a specific domain, are called Domain-specific Languages (DSLs), providing the user with suitable notations and abstractions to describe relevant concepts in a more efficient and understandable way [169, 189, 190]. The developed DSL considers the needs of a specific domain, for example, a problem domain (e.g., production, power generation) or a specific system aspect (e.g., workflows) [191]. The term DSL can be defined as follows.

Domain-specific Language (DSL)

A domain-specific language limits and focuses on a particular application domain, provides appropriate or established notations, and the right level of abstraction to view system solutions in a natural but not overly detailed way [189, 192].

Whether the use of a DSL is actually advantageous compared to a GPL, depends on the use case and must be decided by weighing the pros and cons with respect to the individual application. To give a short overview, the main advantages and disadvantages are briefly listed below.

Advantages:

- Information such as solutions or problems can be expressed clearly and unambiguously by an individually adapted modeling language in terms of expression, semantics, and the degree of abstraction appropriate to the specific domain [193]. The specific focus of the DSL therefore also facilitates domain-specific evaluation and improvement [190, 194].
- The use of domain-specific languages does not require any programming or language skills and is thus easier to learn and use by non-experts (flatter learning curve) [193].
- Due to the properties and structure of the DSL the user can understand the domain as well as the language better and evaluate or modify DSLs easier by themselves [190, 195].
- DSLs and the tools in which they are implemented often enable more efficient/effective automation of, for example, analysis or testing tasks [190, 193].
- By creating models within the DSL, the relevant domain knowledge is documented and makes the DSL an important component of the knowledge management system and enables the associated preservation/reuse of knowledge [190, 193, 196].

Disadvantages:

- Compared to the immediately applicable GPL, a DSL design and implementation generates comparatively high development costs before it can be used (for example, due to the high complexity of the language design) [190, 193]. In addition, cost for its maintenance will be created along the life cycle of the DSL [190]. Furthermore, additional efforts for adaption of a DSL might be created, for example, during a migration.
- Compared to a GPL, it is important for the application of a DSL to define the scope precisely, so all information required for the later description of a system can be expressed and modeled within the DSL [190]. An imprecise scope can lead to necessary adaptations of the DSL and additional effort.

- Despite the fact that the language is easier to learn and understand for users, the individual character of a DSL requires additional training and will generate education costs [190, 193].
- The utilization of DSLs can be hindered by a lack of tool support [193]. In this context, it should also be pointed out that in terms of data exchange a specific DSL also requires, in most cases, a specific matching tool chain. Changes will therefore lead to additional efforts.

To simply summarize, considering the specific advantages and disadvantages, a DSL can play out its advantages in a mainly static environment. In an increasingly dynamic environment however, these advantages are potentially eaten up by the increasing changes that occur in relation to the DSL.

If the use of a DSL is favored, such a language must be designed and defined by the relevant stakeholders. This can be realized, for example, by adapting existing and standardized language profiles such as UML/SysML to the specific applications of the domain of interest (if permitted in the tool) [74, 115, 191]. The actual development of a DSL is a completely independent field of research and will not be considered further in the context of this thesis. A wide variety of sources and articles, such as [197], already address this topic. An enumeration of known DSLs and their fields of application is presented in [190].

As mentioned above, the MBSE tools support the implementation of the MBSE approach by implementing the necessary modeling languages and managing the created models, serving primarily as a source of the entire project knowledge [39]. The tools help to reduce the effort and the number of necessary tools in the development process and increase the efficiency of systems engineering [15]. This is realized, for example, by the automatic adaptation of elements affected by changes, as well as the simplified exchange of models between disciplines and organizations, or the possibility of performing automatic error checks [15]. Due to close interdependency, the selection of the right tool or modeling language influences each other. Therefore, the usage of the different languages depends strongly on the tools available, their support of the individual language, and first and foremost their applicability to the task at hand. In practice, a wide variety of tools is used in connection with MBSE and the creation of architectural content. Common tools are, for example, the paid tools Enterprise Architect [198] or MagicDraw [199] or the open-source tools Capella [200] and Papyrus [201]. Since the selection and best applicability of the tools strongly depends

on the specific use case, the different tools will not be discussed further in the following. The selection must be made individually by the applying stakeholders.

Along the life cycle, artefacts, such as requirements or CAD models, are created using a variety of often non-interoperable tools [135]. The available tool landscape of a company is often influenced by different organizations and disciplines. In order to select suitable tools, for example for the development of a system, classification concepts, as presented in [7], can be used. Furthermore, it should be mentioned that for a continuous application of the MBSE concept and to ensure that the model is complete, correct, and consistent, the different tools have to be connected [39]. Following [25], "[the] challenge is to combine the relevant IT tools into a development environment and to support the interaction of the tools in terms of models, systems, processes and procedures." This is the core task in creating an integrated tool chain. Any further challenges and implementation approaches are not considered in this thesis, as they are not directly relevant to the creation of the architecture framework concept. The linking of tools and the automation of tasks are, for example, described in [202].

6.2 Development Procedure of DSL

The domain-specific language “Industrial Plant Modeling Language – IPML” [203] was developed and implemented during the CrESt research project as described in section 1.3. This language should be used to evaluate several contents within a model-based environment. In addition to content from the architecture framework concept, content developed by other project participants at Siemens AG was implemented in the DSL as well. This additional content will not be discussed further in the following sections and therefore not all parts of the DSL will be presented. The architecture framework concept content actually implemented in the DSL is described in the next section. In this section, the procedure for implementing content within the DSL, like architecture framework content, will be described and the role of the author and other involved stakeholders will be highlighted. Before taking a closer look at the implementation procedure in the following, it should be noted that despite the large number of existing DSLs [190], a new DSL was created in order to fully exploit the strengths of the individual design of processes and content in a DSL. The main tasks, involved stakeholders, and results of the development of the domain-specific language are shown in Figure 70.

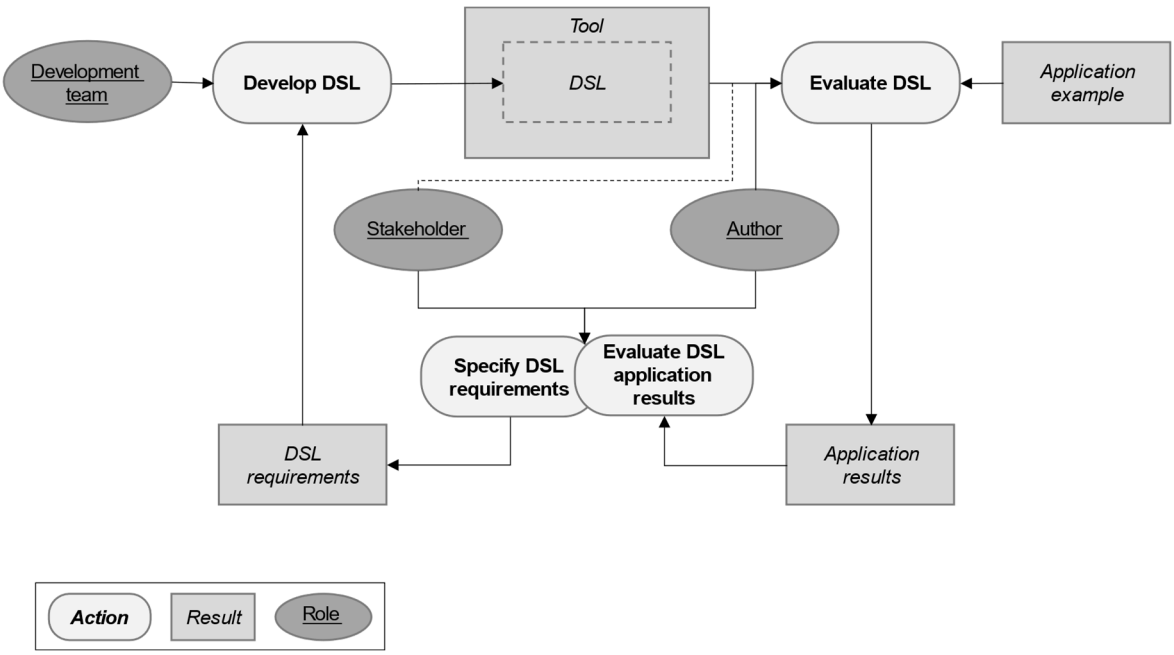


Figure 70: Development of domain-specific language (DSL)

In general, it should be distinguished between two major core areas; the development of the DSL and the evaluation of the DSL. The development of the DSL, which is implemented in a modeling tool, is performed by one or more stakeholders who have the skills to implement a domain-specific language. In the specific case considered here, the actual programming and implementation in the tool was performed by a team of developers from Siemens AG specializing in DSL development, who also participated in the CrESt research project with a focus on modeling and DSLs. This approach enabled the implementation of the DSL in a relatively short time window, with full control over the content, and at the same time avoiding time delays and the need for training and knowledge building with regard to DSL programming. By implementing the DSL by experts, the risk of a possible undesirable development could be reduced to a minimum. Building up the necessary know-how can make sense in the long term if several DSLs are planned to be implemented, which was not the case in this. The DSL was carried out in the MagicDraw modeling tool which will be described in section 6.3. The UML and SysML profiles available in the tool have been customized to realize individual element types and models as well as links between the elements. The development of the DSL itself is based on the defined requirements concerned with desired contents and functionalities of the DSL. These requirements are defined by the author himself as well as other stakeholders and thematically include both the architecture framework concept as well as other topics (as described above). After the implementation, the results, i.e., the DSL, are evaluated and tested by the author and partly by the other stakeholders. For this purpose, parts or all of the components of the production system example (demonstrator) presented in section 1.4 are modeled

with respect to the implemented content. For example, when implementing the possibility of modeling different functionalities, the functionalities of the demonstrator itself are identified and specified in the models provided for this purpose, and the associated interfaces and affected elements, if already implemented, are tested. This prototypical application results in certain specific outcomes that are evaluated by both the author and the other stakeholders with respect to the specified DSL requirements. Based on this evaluation, necessary existing requirements are adapted, further supplemented, or new requirements are defined. Since the implementation of a wide variety of content with different goals specified by different stakeholders was planned in the DSL, a step-by-step approach was chosen in order to be able to implement individual features one after the other without them negatively affecting each other. This means that the specification-definition-evaluation cycle is run through for the implementation of the individual features, if necessary, several times for each feature, until the content is implemented in the DSL in a complete and usable manner.

6.3 MagicDraw Tool Introduction

As already indicated in section 6.1, there are several commercial and freely available modeling tools on the market that can be used in a model-based engineering environment. These tools can be used with respect to DSLs and follow different approaches for modeling contents, such as text-based, graphics-based or mixed-notation concepts [169]. Text-based approaches are used, for example, in tools such as MontiCore [204] or Xtext [205], graphic-based approaches, for example, in Enterprise Architect [198] or MagicDraw [199], and mixed approaches, for example, in mbeddr [206] [169]. The creation of the DSL also varies between tools and ranges from scratch definition to customization of general-purpose languages, such as UML [169]. An individual choice of the most suitable DSL tool should be made depending, on among others, the intended goal, the initial situation and the scope of the DSL. In addition, constraints of relevant stakeholders or organizations can have a major influence on the selection, which often leads to a situation in which a compromise must be found between the supposedly optimal solution and the actual environment in which the tool shall be applied. Within the CrESt research project and in relation to the existing environment at Siemens AG, the MagicDraw tool was selected for the realization of the IPML, taking economic considerations, availability of tools, and accessible knowledge in the organization into account. Therefore, within this thesis, MagicDraw was also used as the tool of choice regarding the application and evaluation of the specified architecture framework concept.

General information with respect to MagicDraw, its features, and contents can be looked up on the web page of the developer [199] or in the tool manual in detail [207]. In the following, the interface of the tool, which is visible for the stakeholder and used in the modeling, is briefly described in order to give a first impression of the tool and on a possible mode of operation in the tool. After starting the program, the welcome screen appears in which, among others, projects can be managed [207]. In the manage projects area a new architecture project can be created, or an existing project can be opened. Regardless of whether, for example, a UML-, SysML-, or DSL-based project is opened, the main window shown in Figure 71 is displayed to the concerned stakeholders. This window can be divided into six main areas, which fulfill different tasks during the modeling of a system. These areas are the main menu, main toolbars, model browser, diagram toolbars, diagram pallet, and the diagram pane [207].

In the main menu, various predefined and modifiable sub-menus display possible commands in the form of different items [207]. Sub-menus, like the file menu or the diagram menu, display, for example, commands such as "open project" or "save project" or commands such as "create diagram" or "customize" [207]. The selection of a command leads to connected actions within the program. For example, using the "delete" command will delete a selected content/item [207]. Frequently performed tasks and the commands needed for them are displayed as shortcuts in the so-called toolbars and assist at increase the editing speed [207]. As already mentioned above it can be distinguished between the main toolbars and the diagram toolbars. In the main toolbars frequently used commands are arranged in relation to, among others, file, project, diagram, and validation [207]. The diagram toolbars contain frequently used commands related to the creation and handling of diagrams. These toolbars include, for example, commands related to navigation, layout, and edit [207]. Detailed information about available commands of the toolbars and about customization of menus can be found in the MagicDraw manual.

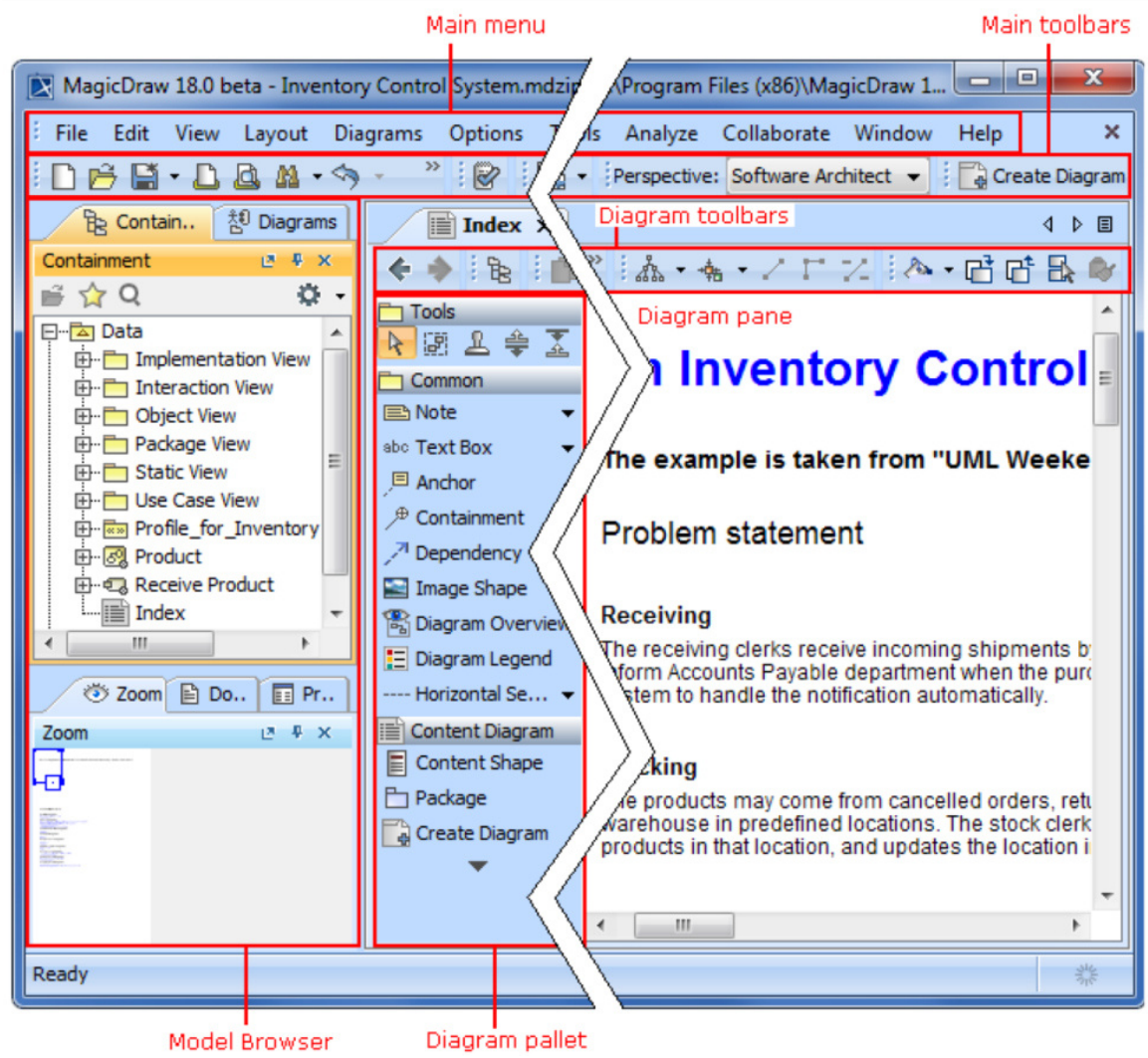


Figure 71: Contents and structure of main MagicDraw window [207]

Another core element, which the stakeholder(s) repeatedly refer to during modeling, is the model browser, which displays and visually depicts all model elements that have already been defined and will be defined [207]. The model browser structures the available data in hierarchical form and is used to manage this data [207]. In addition, the model browser can also be used as an alternative to the menus and toolbars [207]. Among the possible operations that can be performed via the model browser are the creation, editing and deletion of models, the management of diagrams, and the sorting of model elements [207]. Also, the model browser can be divided into further sub-components as shown in Figure 71. Those sub-components thematically illuminate related topics such as model extension, search results, and documentation [207]. More detailed information on the model browser can also be found in the MagicDraw manual.

As described by using the menu, the toolbars, or the model browser, diagrams and elements can be created, for example by command or drag and drop. As soon as a

corresponding content-sensitive diagram has been opened or selected, possible options available in the form of specific elements or actions are displayed in the diagram pallet [207]. The created diagram appears in the diagram pane [207]. Using the model browser or the diagram pallet, corresponding diagram elements can then be created and related in the diagram pane. The diagram pane displays all with relationship to the selected model created and deployed contents.

In summary, it can be stated that regardless of the project (for example, GPL or DSL based), the same interfaces components are used for modeling a project. In simple terms, as described above, the corresponding diagrams and elements are defined via the interface, set in relation to each other, and managed in the tool.

In relation to this thesis and to the implementation of the DSL some individualities should be mentioned, which are briefly discussed below. If a DSL has been implemented in the tool and is used in a selected project, the interface window and the components described above do not change structurally and role-wise, but they might represent specific contents. Naturally, for example, also in a UML-based project specific contents are indicated. At this point however specific means that beside the classical elements of the UML, on which the DSL is based, also customized elements and diagrams are specified. For example, when opening a new DSL-based project the model browser might already display a specific hierarchy tailored to the modeling of a particular system, which might already be filled with predefined diagrams as well as elements and only allows certain content to be created by the stakeholders for the subsequent definition. These specific customizations might be reflected across the entire interface. Those customizations might range from specific diagram names and types up to pre-selected objects in the diagram pallet. These possible freedoms of a DSL are ideally and depending on the intended goal designed in such a way that the stakeholder(s) does not have the feeling that they are missing mandatory objects for modeling their system and can concentrate primarily on the creation of the content. Such considerations have ideally been made beforehand and are already implemented in the DSL.

6.4 Scope of DSL with respect to Architecture Framework Concept

Before the implemented DSL is presented in the following section, this section briefly discusses which components of the specified architecture framework concept were incorporated into the DSL. The restriction of the implementation had different reasons, which were mainly in the budgetary, temporal, and tool-specific technical range. Therefore, a prioritization was performed, and the components considered most

relevant were implemented. In the following, the considerations and aspects that led to the present version of the DSL with respect to the content of the architecture framework concept are briefly described.

In simple terms, the architecture framework concept consists of three related areas that could be considered for implementation in the DSL. These are the structure of the framework, determining the general approach through the defined dependencies and viewpoints/views, the predefined content element types that fit into this structure, and the methodical approach for the definition, change, and transition of content. The methods make use of the structure and predefined element types as well as external inputs.

Ideally, all these elements should be integrated into the DSL in a complete manner, but as mentioned above, this was not possible within the scope of this thesis and the research project. For this reason, the implementation of the content was prioritized on the basis of the causal relationships between the areas of the framework. As indicated, the framework is based on a fundamental structure (domain to technical viewpoint - see chapter 5.4.2), into which relevant views and associated contents are then defined. Therefore, in a first main implementation step, this basic structure, the associated viewpoints and views, as well as pre-defined elements and model types were defined. Additionally, first basic relationships between the element/model types were implemented. In a second main implementation step, this structure was broadened by relevant content element types and the interrelationships between elements were further specified to enable continuous tracing of, for example, requirements to functions, logical elements, and technical solutions. In a third implementation step, the static considerations, for example, requirement tables, were extended by dynamic modeling, for instance, in the form of activity diagrams, so that both the static and the dynamic behavior can be modeled, both of which are relevant for the creation of a comprehensive architecture description. After a state of implementation was reached in which a system architecture description can be created with the specified elements of the DSL, in a next implementation step, a concept for the creation of several such identical structures was defined. This was realized to represent the granularity layers within a system and as defined within the architecture framework concept. In addition, specific relationships were created to trace between contents of the different granularity layers. This enables, for example, to show how requirements at the top layer are detailed at a higher granularity layer. Thus, the architecture framework concept specified in section 5.4 is largely mapped in the DSL with regard to the structural and content components. Different versions of the overall DSL were documented at the time of the CrESt research project in [169, 208].

With respect to the methodological components of the architecture framework concept, only method related contents were added to the DSL. First and in parallel to the implementation described above, a documentation was created within the tool, in which the individual element types, model types, relationships, and basic procedures were described. In the description, the role of the element/model types, their meaning, and their use in the framework were highlighted in order to be able to support the relevant stakeholders in applying the specified architecture framework method and to be able to store relevant knowledge accessibly in the tool. Second, the ability to share and create versions of a model, as well as the ability to compare model content, was utilized to support the methodological approach of the framework by leveraging existing reference architect descriptions in a copy and modify approach for creating system architecture content. All other methodological contents of the architecture framework concept have not been implemented due to the setting in which the development took place, the associated time and budget constraints, as well as to technical limitations of the tool.

The components of the architecture framework concept above introduced mainly summarized the content implemented within the DSL. Although not all components of the framework could be implemented, the required content for application was implemented based on the causal relationship of the elements. By defining structure and content, the DSL can in principle be used without implementing any methodology, given an existing domain and modeling knowledge of the stakeholder utilizing the DSL. Achieving this scope was the main goal of the DSL development with respect to time and budget constraints. This implementation order was chosen because the specified architecture framework concept method can be used in combination with the DSL to define architecture descriptions even without explicit implementation of the available structure and content requirements. Otherwise, i.e., in case of a methodology without structure and contents, a much higher domain knowledge would be necessary and the specification of reference and system architecture contents would be more difficult.

For the sake of completeness, it should be mentioned at this point that for the implementation of the content described above, specific adaptations were made to the UML2 profiles on which the DSL is based [169]. These adaptations are not directly visible and include, among other things, the creation and assignment of specific elements, icons, associated properties, and elements for structuring (for example, packages) [209]. Finally, it should be noted that each of the results were implemented step by step in the DSL by using the process described in section 6.2.

6.5 Example of Implementation of Architecture Framework Concept

As described in the previous sections, a domain-specific language (DSL) for modeling industrial plants (IPML) was implemented in the MagicDraw modeling tool and the exact implementation scope was given. In the context of the creation of the IPML, contents which can be mapped onto the architecture framework concept were created. In this section, the implementation is exemplified by a set of images from the tool.

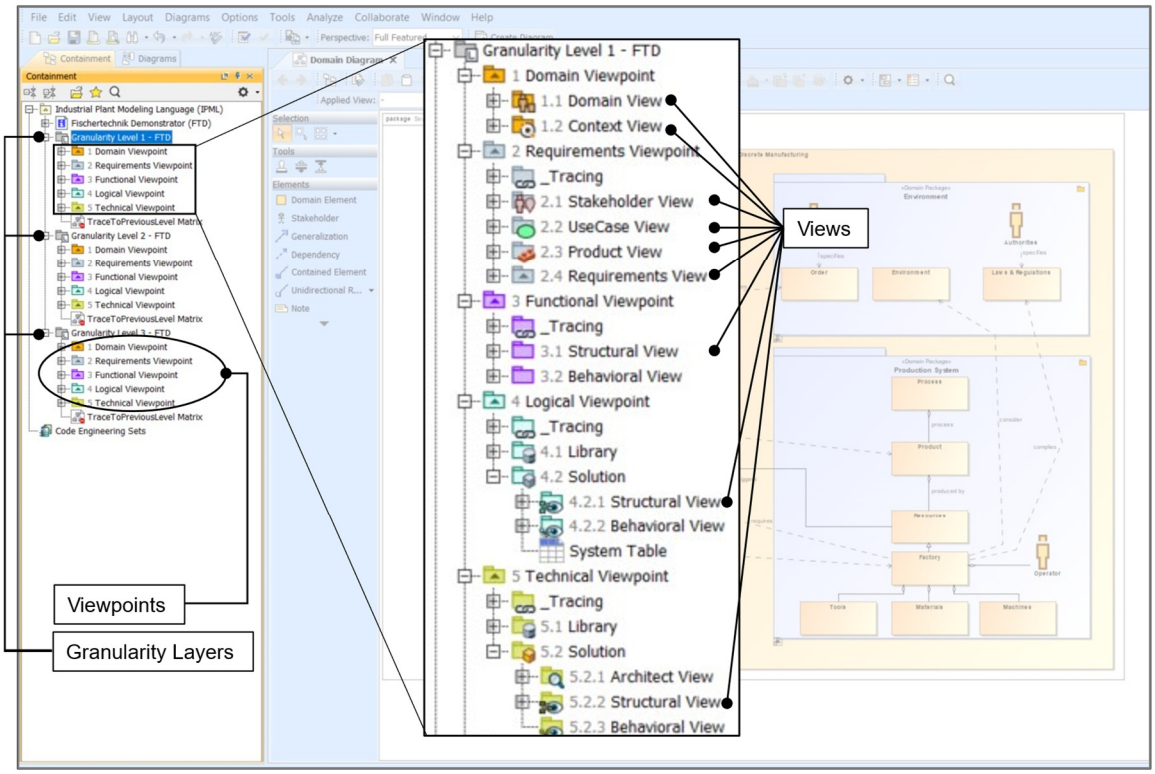


Figure 72: MagicDraw main window with specific contents utilized within architecture framework concept

Figure 72 shows the MagicDraw main window as in Figure 71, but with the specified contents utilized within the architecture framework concept. Within the figure the structure, with viewpoints and granularity layers, as well as the specified views are displayed.

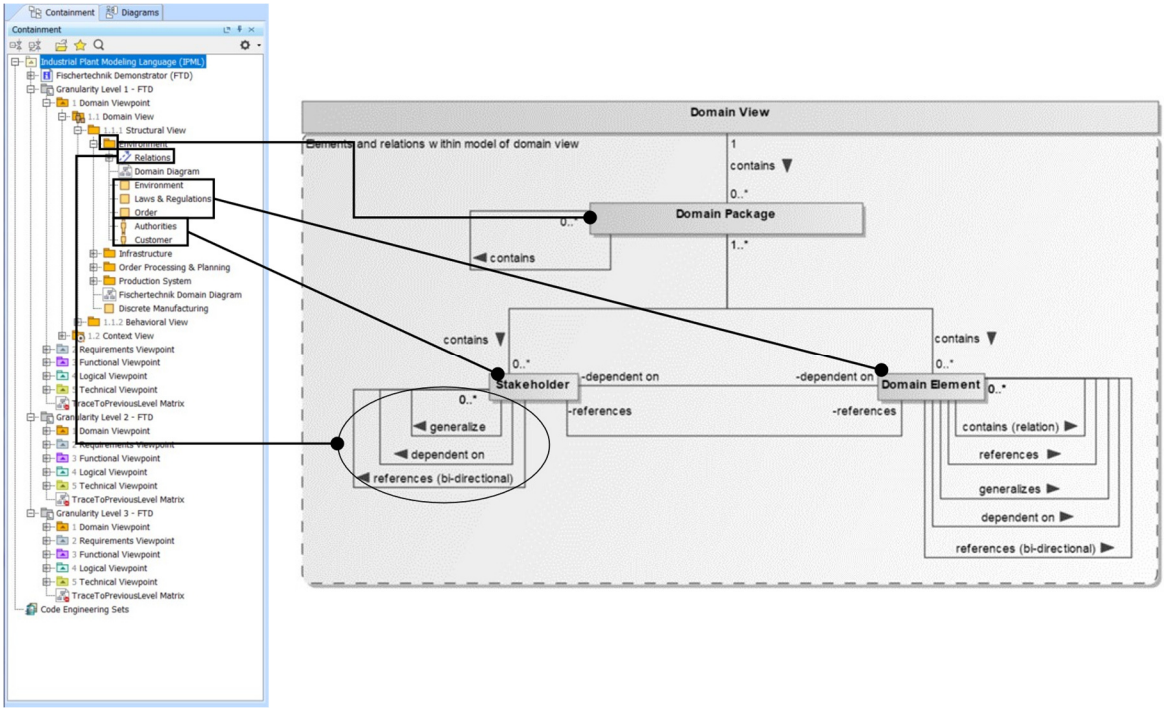


Figure 73: Exemplary representation of content element types - adapted from [169]

Figure 73 shows an example of how the specified contents of the individual views are stored in the model browser. For the domain view, for example, these are the different domain elements and domain stakeholders (here generally named like the appropriate element types). These can be created, for example, by using the model browser or in the corresponding domain model diagram, as instances of specific stakeholders. The specific stakeholders would then also be mapped in the model browser or in the diagram in which they were defined. Figure 74 shows a diagram of a domain view and exemplarily defines how relationships between stakeholders and elements can be modeled. The predefined element types are dragged and dropped from the diagram pallet into the diagram pane and instantiated. At this point, the individually tailored structure of the DSL becomes clear in comparison to a GPL, since only the elements and relationships required for the creation of the model are available. This content changes automatically depending on the view and model.

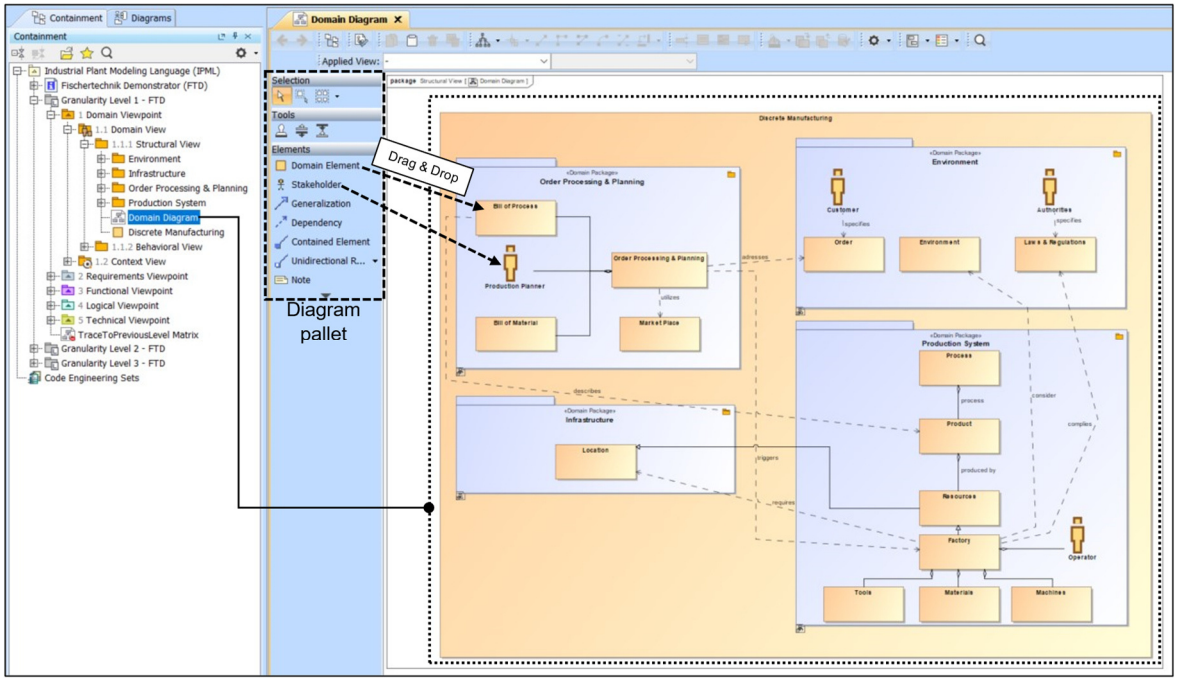


Figure 74: Example for modeling a domain view

If new content has been defined, the content as described above is allocated automatically to the considered view and shown in the model browser. Figure 75 shows the model browser before and after the definition of additional content. In the example interests within the stakeholder need view have been specified.

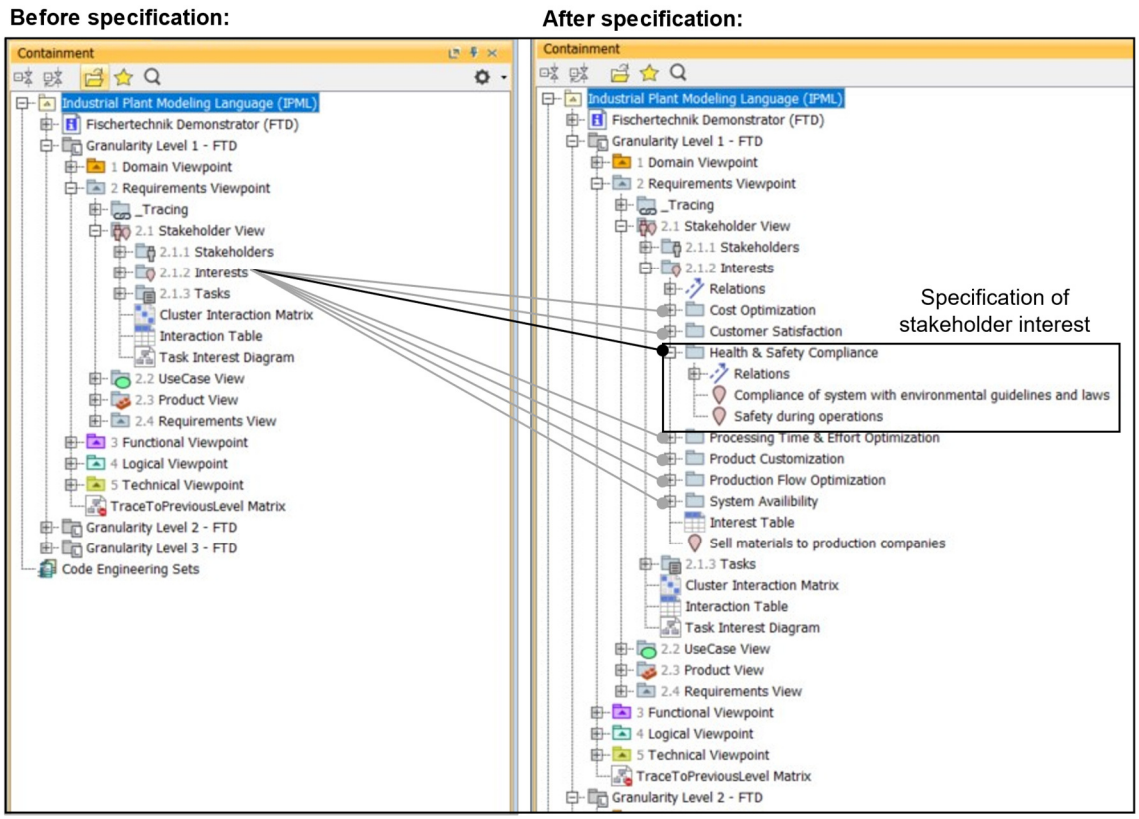


Figure 75: Representation of content within model browser - adapted from [169]

After the structure and the content element types used during the application of the architectural framework concepts have been presented in the previous figures, it shall be shown in the following how the individual defined contents are set in relation to each other in the tool. This enables, for example, traceability and the possibility of automatic identification of relevant inputs for further definition steps. To provide an example by regarding Figure 76, the individual views of the requirement viewpoint, and the content they contain can be set in relation to each other. In this example this is constituted via matrices in which use cases and different requirements that can result from these use cases are compared. In the matrix, the stakeholder utilizing the tool for the definition of an architecture description can then specify which requirement realizes which use case.

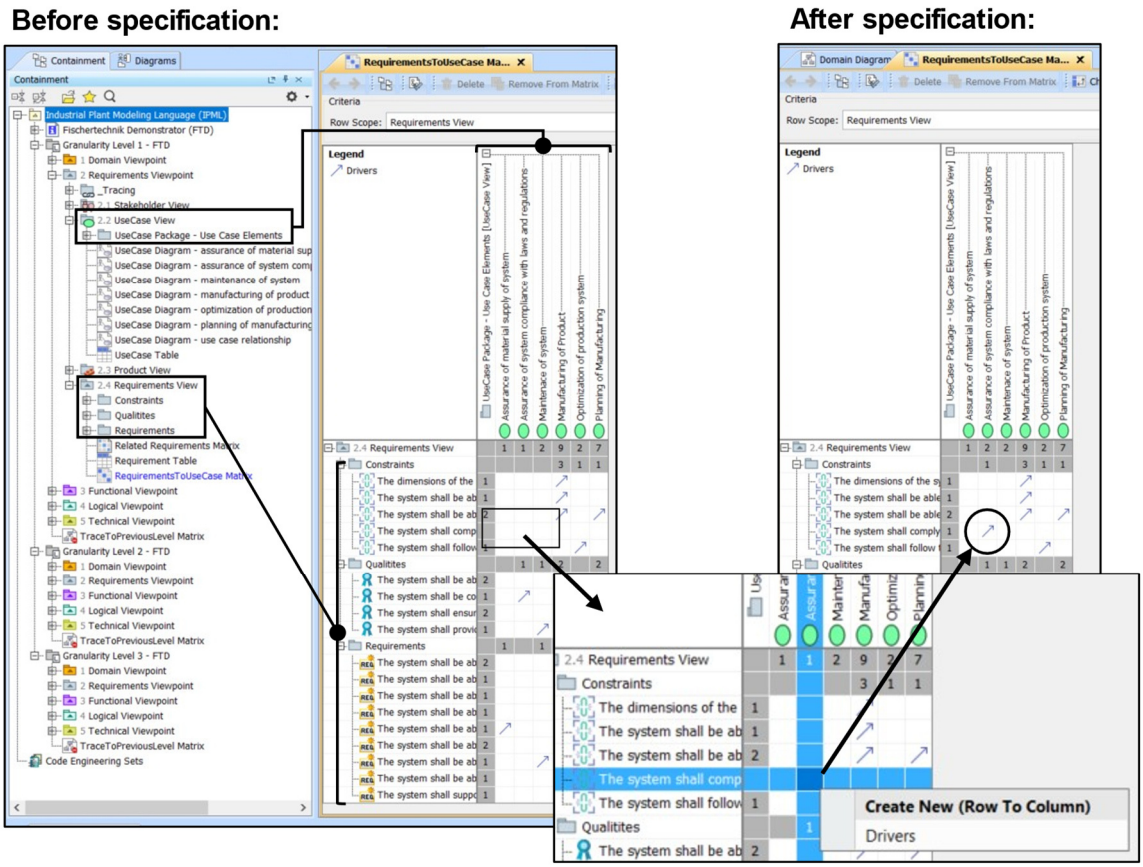


Figure 76: Exemplary definition of relations between contents of different views – adapted from [169]

The implementation of the structure and element types presented in chapter 6 will be used in the following to test the application of the architecture framework concept for the definition of a system architecture description based on a reference architecture description. The corresponding methodologies have not been implemented for the reasons mentioned above but can still be applied and the results generated in the tool.

7 Prototypical Application of Architecture Framework Concept

In order to evaluate the contents presented in the thesis, in addition to publications (see Bibliography) as well as presentations and discussions at selected conferences, the developed contents were discussed with relevant experts from industry and research. These include, among others, the supervising professor of this thesis, Prof. Dr.-Ing. habil. Arndt Lüder, from the Otto-von-Guericke University Magdeburg, as well as experts from Siemens AG - Corporate Technology department and the CrESt research project. For further validation of the developed architecture framework concept (chapter 5) which was integrated into a model-based engineering tool in the form of a DSL (chapter 6), a prototypical application is carried out within this chapter. The goal is to evaluate the concept of the elaborated architecture framework for the definition of a system architecture description based on a reference architecture description.

Section 7.1 describes the procedure for the prototypical application and the relevant contents, methodologies, and structural components. In sections 1.4 and 7.2 the application example as well as the reference architecture used for the evaluation are described. Subsequently, section 7.3 shows some examples of results created during the utilization of the architecture framework concept. Section 7.4 summarizes the results of this application and gives an assessment of the results. For an overview of content and relation to other chapters as mentioned above see Figure 77.

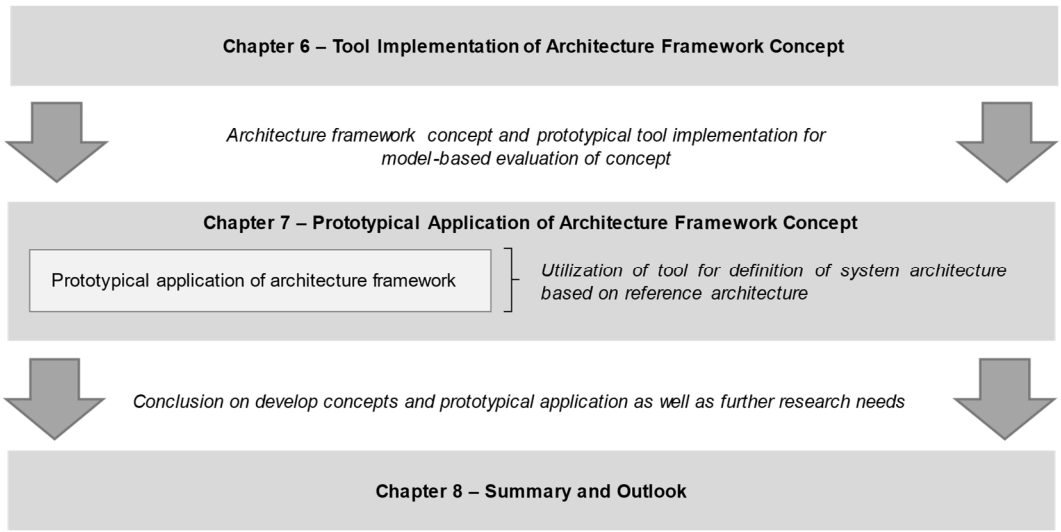


Figure 77: Overview of content of chapter 7

7.1 Application/Evaluation Objectives and Procedure

For the evaluation of the developed architecture framework concept, the framework is prototypically applied, and a system architecture description is derived from an existing reference architecture description. For this purpose, the structural and content-related components of the architecture framework concept presented in chapter 5 are utilized in the form of a model-based systems engineering approach as described in chapter 6. The methodological aspects of the architecture framework concept will be applied manually as specified and the results will be represented within the tool. For the evaluation of the architecture framework concept the reference architecture described in sections 3.3.3 and 7.2 will be used. To derive a specific system architecture description, the requirements towards that system must be clearly described. Therefore, another input to be considered within this evaluation is a representative set of requirements of the application example presented in section 1.4. This application example is acting as a specific system representation within this evaluation. Based on those two inputs the specified architecture framework concept is prototypically applied. The created system architecture description based on the reference architecture description is evaluated with respect to the intended use of the architecture framework concept as well as to the specified research questions. The described evaluation procedure is shown in a simplified fashion in Figure 78.

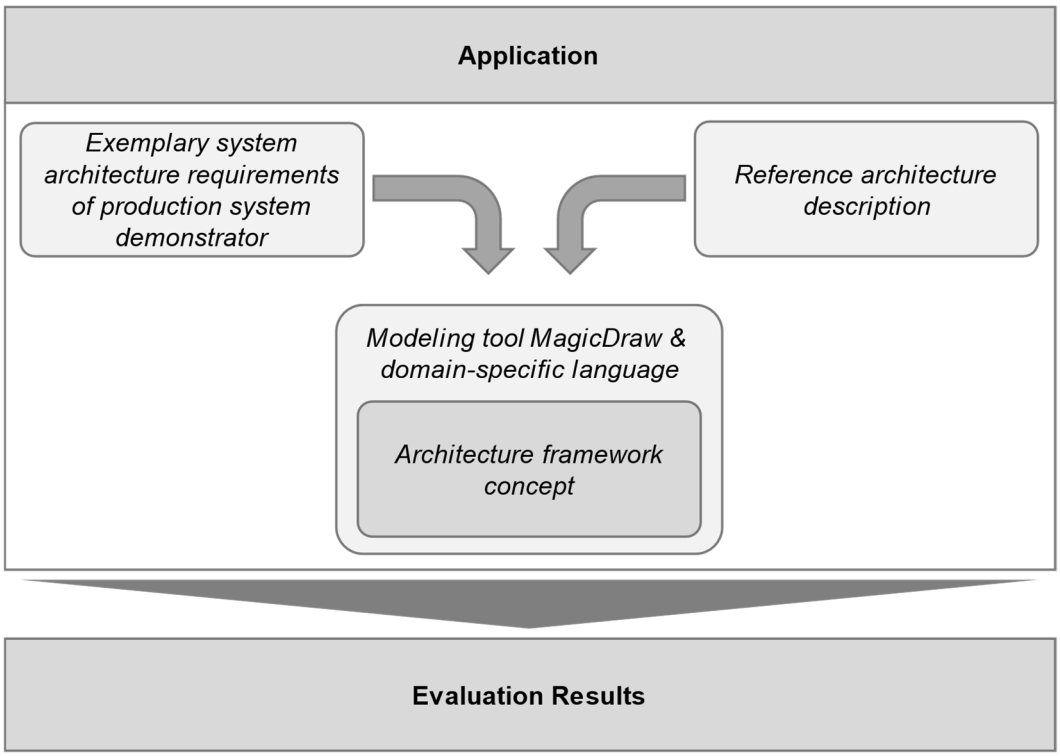


Figure 78: Overview of prototypical application of architecture framework concept for evaluation

Furthermore, it should be noted and differentiated at this point that the main goal of the evaluation is not to assess the direct result of the architecture framework concept, i.e., the derived system architecture description, but rather to analyze whether the specified architecture framework concept fulfills its goal of intervening in a supportive manner in the definition process when deriving specific contents from a reference architecture. Thus, the focus is more on the procedure than on detailed contents of the architecture. The derived architecture results are still considered in the evaluation as it also allows conclusions to be drawn about the framework itself, but it shall be mentioned, that results can be better evaluated by using processes actually designed for this purpose, such as the Architecture Tradeoff Analysis Method (ATAM) [210]. The nine-step method [61, 210], which originates from software engineering, can also be used in an adapted form for systems engineering [38] and focuses mainly on "[...] the goals of the system and the business around the system, the quality requirements that meets the goals, and finally the architecture that satisfies the quality requirements" [61]. Since this kind of evaluation does not take the predominant role in the thesis, the process is not further regarded and can be looked up, for example, in [61] and [210] for more details.

7.2 Utilized Reference Architecture

For the prototypical evaluation of the architecture framework concept, the reference architecture defined in [35] and presented in section 3.3.3 of the state of the art chapter is used. There are several reasons for using exactly this reference architecture in the context of the prototypical evaluation of the architecture framework concept, which will be briefly explained in the following.

The first assumption made during the creation of the architecture framework concept is that the inherent focus of the reference architecture and of the system of interest must share a critical amount of content and rely on similar structure and content types. The focus of the introduced reference architecture is not exactly on the scope of discrete manufacturing, but on a topic directly related, the adaptable and flexible factory. If due to the higher degree of abstraction of the reference architecture, the scope of the reference architecture includes the content scope of the system to be derived, the contents of the reference architecture can be used as a template for this project. In addition, due to the common relationship of both the architecture framework concept and the reference architecture to the CrESt research project, the reference architecture is also based on the SPES_XT modeling framework. Therefore, the

framework as well as the reference architecture to be used in it have the same basic structure.

Second, the contents of the reference architecture should be available to the author performing the evaluation. Since the contents of the reference architecture were developed as part of the CrESt research project [32], they are available in their entirety and can be used. In addition, the content was specified and implemented based on the SPES-XT framework developed in previous projects [33, 34] and utilized in CrESt [32, 35] and the DSL presented. Thus, the contents are available in a model-based form in the modeling tool. Furthermore, the reference architecture descriptions are based on the same structure and use the same content element and relationship types that will be used for modeling of the system architecture, allowing direct comparability and application of the framework methodologies developed. This equality was made as assumptions for the use of the architecture framework in section 5.2.

The points mentioned above, are the main reasons for the utilization of the described reference architecture within the prototypical evaluation of the architecture framework. This means that all requirements that were previously defined are met.

7.3 Prototypical Application of Architecture Framework Concept

This section presents an exemplary excerpt from the prototypical application of the architecture framework concept for the definition of a specific system architecture description of the production system application example. The system architecture description is based on the reference architecture description developed in the CrESt research project. First, the views as well as associated contents of the domain and the requirement viewpoint on the granularity layer one of the system of interest are presented. As mentioned with respect to the architecture framework concept, it is assumed that the problem space of the system of interest is defined separately before the transition. This description of the problem space, domain, and requirement viewpoint is used to analyze whether the reference architecture is suitable for the definition of the system of interest by comparison between required and available content. Additionally, this forms the basis for the transition of the contents from the predefined reference architecture description for the definition of the solution space of the system of interest. The transition of the logical view is described in detail below. In order to complete the representation of the production system application example on GL1, the results of the definition for the functional and technical view are shown as well. The contents considered for the prototypical application are highlighted in Figure 79. The consideration of the problem space as well as the definition of the contents of

the associated viewpoints and views on granularity layer one of the system of interest are shown as a green-gray striped box. Those considerations are made in section 7.3.1. The transition between reference and system architecture, to define the solution space of the system of interest, considers the logical view on GL1 as an example. The light blue box describes the contents of the reference architecture that can be reused as input. The dark blue box describes the result of the transition in the architectural description of the system of interest. The orange boxes of the functional viewpoint represent the inputs for the definition directly considered. The light green box represents the results of the shown transition for the technical view on GL1. The examples for the transition are documented in section 7.3.2.

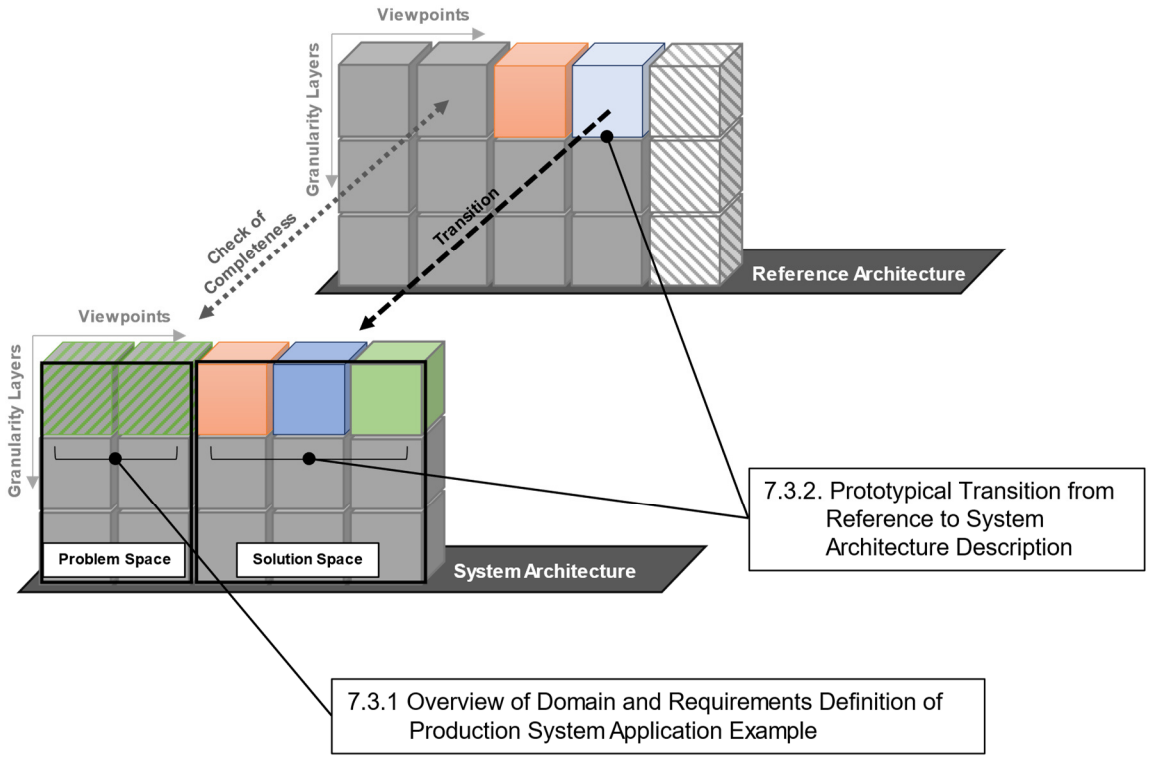


Figure 79: Overview of application content exemplarily shown in this section

Based on the prototypical evaluation of the production system application example and the obtained results, the evaluation of the architecture framework concept is conducted from which the final conclusion is drawn.

7.3.1 Overview of Domain and Requirements Definition of Production System Application Example

As defined in connection with the development of the architecture framework concept in chapter 5, the problem space of the system of interest is defined before the transition within the associated views on the different granularity layers using the architecture framework concept. This definition is described in more detail for the

domain view of the system of interest by proposing, the production system application example. For the remaining views of the problem space the created results are introduced (context view of domain viewpoint as well as stakeholder need view, use case view, product view, and requirement view of requirement viewpoint). All examples shown describe the system of interest on granularity layer one.

Utilizing the architecture framework concept and in particular the architecture content creation method (see Figure 53), the scope of the desired domain description on granularity layer one is defined. In a second step, based on the scoping of the first step, the predefined element-types within the tool were used to model all instances of elements relevant within the domain, for example, the production system itself and the surrounding environment. Additionally, the defined elements were related among each other utilizing the relations available. After the domain of the production system application example has been defined, in a third step, the result was investigated. The specified domain view is shown in Figure 80.

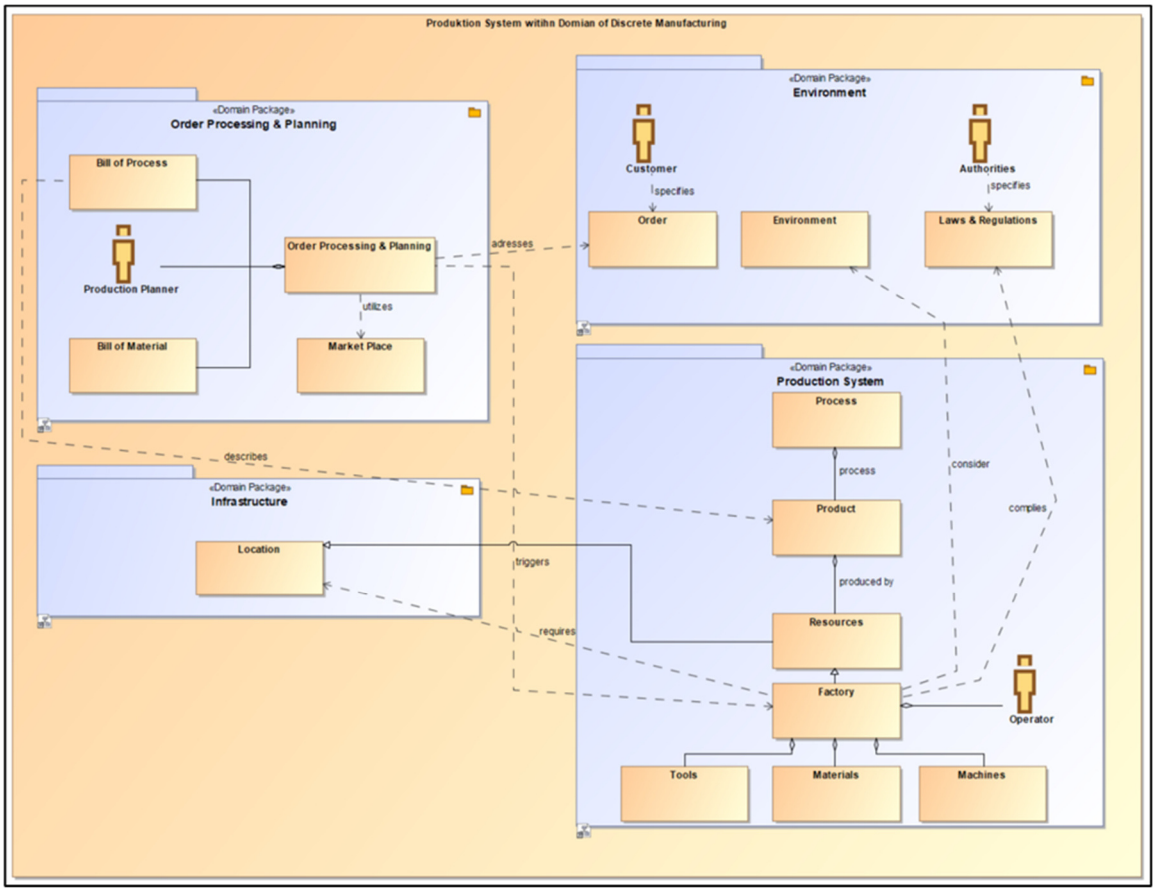


Figure 80: Domain view - model of the considered domain on GL1

Based on the contents of the domain view, the system is delimited from the relevant context and the relevant interfaces are defined. In the context view shown in Figure 81, the cylinder head manufacturing example represents the system of interest, with interfaces to other elements relevant in a company, such as the warehouse or order processing, but also to the surrounding infrastructure and relevant standards as well as applicable laws. With the general consideration of the domain in the domain view and the more detailed delimitation of the system from its environment in the context view, the domain viewpoint is described completely on granularity layer one and the contents of the following requirement viewpoint can be defined.

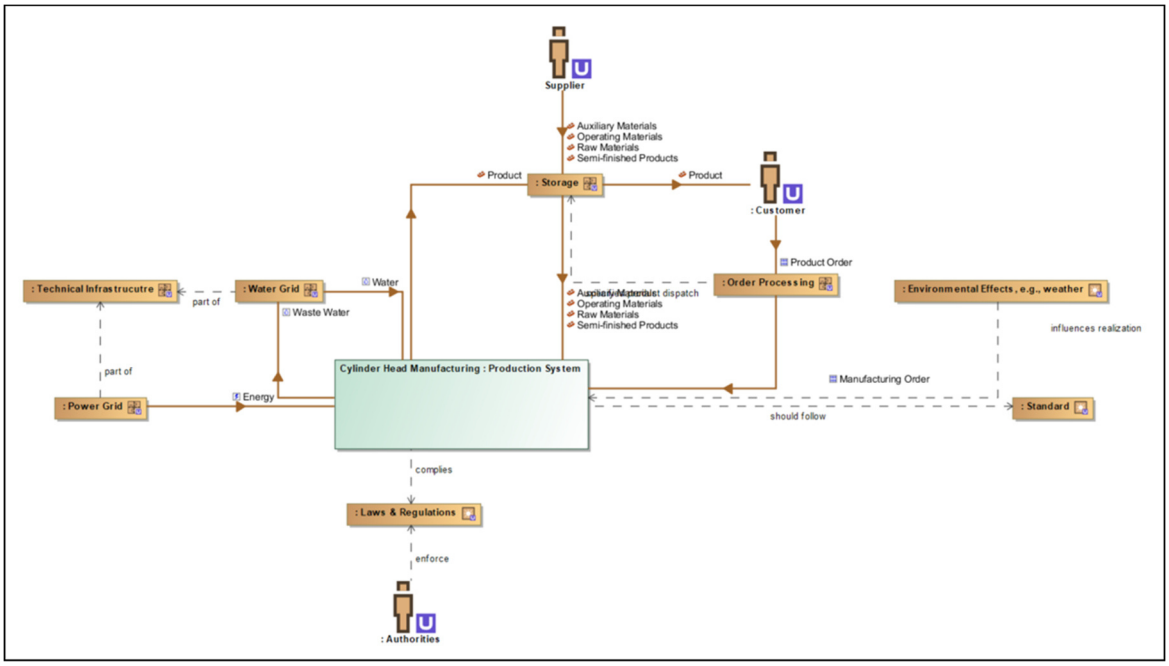


Figure 81: Context view - model of the system of interest context on GL1

Based on the results of the domain viewpoint, the contents of the stakeholder need view can be defined, taking the architecture framework concept into account. An excerpt of the results is shown as an example in Figure 82. As already described in section 5.4.3.4, each stakeholder has tasks and interests through which the respective needs are expressed. For example, the stakeholder "operator" can be assigned the task "execute manufacturing operations" and the interest "machine and tools availability". In this way, all relevant stakeholders and the corresponding tasks and interests, which the stakeholders can also share, should be defined. The consideration of the various stakeholders with influence on the system of interest is an integral part of a comprehensive specification of requirements to the system.

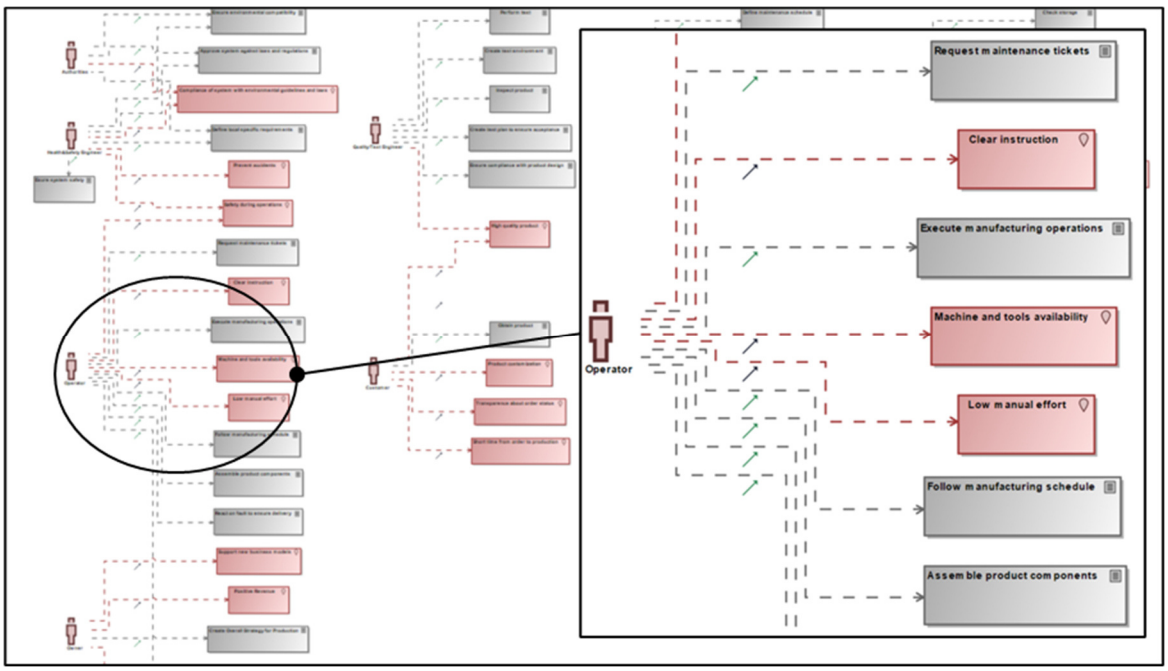


Figure 82: Stakeholder need view - Model of stakeholder, their interest, and tasks on GL1

The defined stakeholders are then connected to the system via possible use cases, also taking into account the relevant context/domain. The use cases describe different scenarios in relation to the system of interest and consider which stakeholders could be involved. These relationships also result from the role, interests, and tasks of the respective stakeholders. For example, in addition to the production system in the use case "manufacturing of products", stakeholders such as the operator, maintenance personnel, or the health & safety engineer may be involved. An example of the results defined in the use case view is shown in Figure 83.

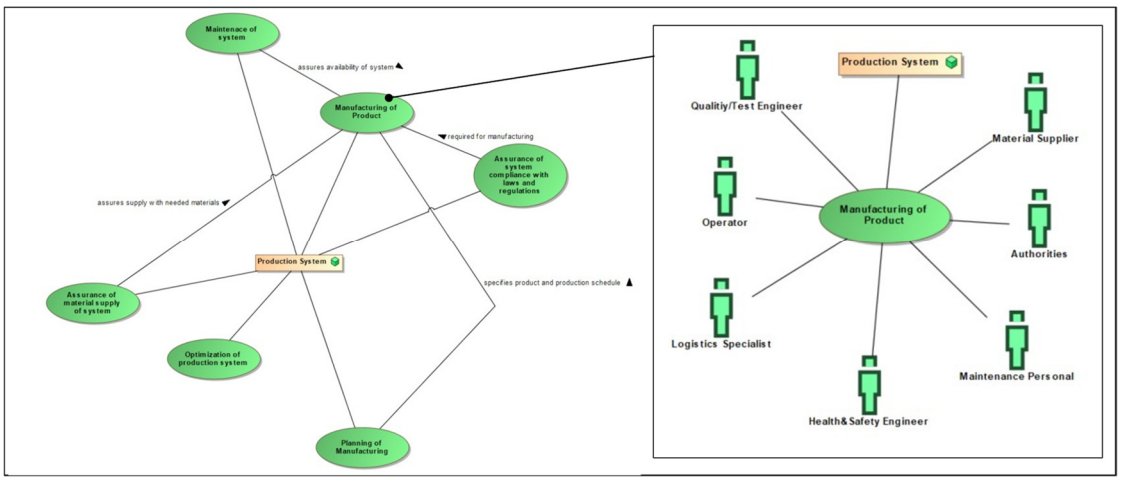


Figure 83: Use case view - model of use cases on GL1

In addition to the stakeholders and the use cases, the product to be manufactured and the associated manufacturing processes must always be considered when describing a production system as system of interest. The manufacturing process for the cylinder head already described in Figure 5 has to be defined within the product view of the requirement viewpoint. The results are shown in Figure 84. Within the figure the relevant components of the product and the individual process steps are modeled, for example, the milling of the contour of the cylinder head or the test assembly. Specific product requirements can then be derived from this in a next step.

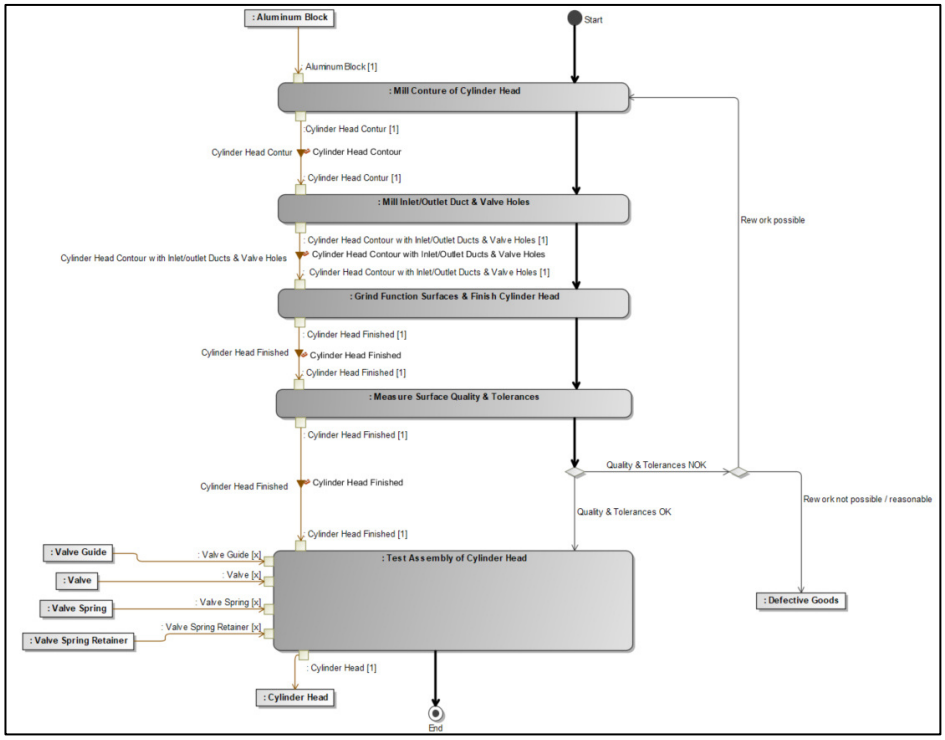


Figure 84: Product view - representation of product and production process on GL1

Based on the contents of the views of the requirement viewpoint, the resulting requirements, qualities, and constraints are defined in the requirement view in relation to the system of interest. In Figure 85 these are exemplarily represented for the modeled example alongside the relationships with the previous considered elements, such as use cases or stakeholder tasks. Thus, at granularity layer one, requirements such as "the system shall be able to manufacture a product with different specifications within a defined product spectrum" or "the system shall be able to handle materials" or constraints such as "the system shall comply with laws and regulations" arise.

#	Name	Type	Kind	Drivers
1	The dimensions of the system shall not exceed the available installation space	Constraint [Class]	System	Manufacturing of Product
2	The system shall be able to access, store and process order data	Functional Requirement [Class]	System	Manufacturing of Product Planning of Manufacturing
3	The system shall be able to determine the current status of the production	Functional Requirement [Class]	System	Planning of Manufacturing Define manufacturing schedule <--> Cost El
4	The system shall be able to execute control commands	Functional Requirement [Class]	System	Manufacturing of Product Execute manufacturing operations <--> Pos
5	The system shall be able to follow and execute production related instructions	Functional Requirement [Class]	System	Manufacturing of Product Execute manufacturing operations <--> Pos
6	The system shall be able to handle materials	Functional Requirement [Class]	System	Assurance of material supply of system Execute manufacturing operations <--> Pos
7	The system shall be able to handle products of different shapes	Constraint [Class]	System	Manufacturing of Product Execute manufacturing operations <--> Pos
8	The system shall be able to manufacture a product with different specifications within a defined product spectrum	Functional Requirement [Class]	System	Manufacturing of Product Planning of Manufacturing Execute manufacturing operations <--> Pos
9	The system shall be able to produce different variants of a product within a defined product spectrum	Constraint [Class]	System	Manufacturing of Product Planning of Manufacturing Execute manufacturing operations <--> Pos
10	The system shall be able to provide customized products	Req. Quality [Class]	Stakeholder	Manufacturing of Product Planning of Manufacturing Execute manufacturing operations <--> Pos
11	The system shall be able to spupport maintenance of the overall system and its sub-systems	Functional Requirement [Class]	t.b.d.	Maintenace of system
12	The system shall be able to support the planning of a product	Functional Requirement [Class]	System	Planning of Manufacturing Define manufacturing schedule <--> Cost El Define production tasks <--> Cost Effective
13	The system shall be compliant with environmental guidelines and laws	Req. Quality [Class]	System	Assurance of system compliance with laws a
14	The system shall comply with laws and regulations	Constraint [Class]	Business	Assurance of system compliance with laws a
15	The system shall ensure short time from order to production	Req. Quality [Class]	Business	Manufacturing of Product Planning of Manufacturing
16	The system shall follow the overall production strategy	Constraint [Class]	t.b.d.	Create Overall Strategy for Production <--> Optimization of production system
17	The system shall provide equipment maintenance infromation based on machinery data trends	Req. Quality [Class]	System	Maintenace of system Define maintenance schedule <--> Cost Eff
18	The system shall support the optimization of itself and of its sub-systems	Functional Requirement [Class]	t.b.d.	Optimization of production system Optimize manufacturing <--> Cost Effective

Figure 85: Requirement view - model of derived requirements, qualities, and constraints on GL1

With the definition of all contents in the domain view, context view, stakeholder need view, use case view, product view, and requirement view, all views within the domain and requirement viewpoint are described on granularity layer one and thus the relevant problem space for the system of interest is defined on GL1. After the separate and complete definition of the problem space, the transition method can be used to check the contents of the problem space for completeness, by comparing them to the reference architecture description. Based on these results, relevant contents for the solution space can be derived from the reference architecture description using the architecture framework concept and it can be verified, if the RA is suitable for a transition. In the next section the procedure and the results of the solution space are exemplarily described.

7.3.2 Prototypical Transition from Reference to System Architecture Description

As an input for the definition of the logical view of the system of interest on granularity layer one, the relevant functions need to be defined on the same granularity layer of the system of interest. Resulting from the consideration of the contents of the previous views within the architecture content creation method, the relevant functions on GL1 are defined, for example, “produce product portfolio” or “schedule material for production”. The functions and the relationship to the requirement fulfilled by the functions are shown in Figure 86.

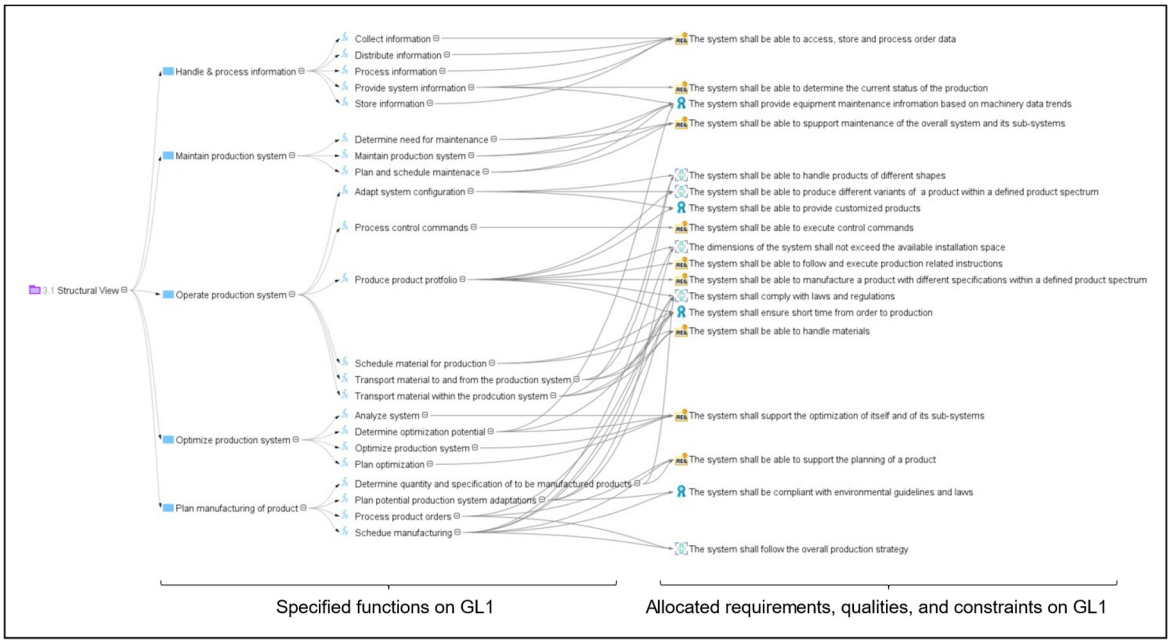


Figure 86: Functional view - specified functions and allocated content of requirement view on GL1

Utilizing the architecture framework concept, Figure 87 shows, with reference to Figure 65 of chapter 5.5.2.3, how the contents of the logical view of the production system application example can be derived from the CrESt reference architecture. In the first step, the customization/creation consequence is evaluated. This is illustrated in Figure 87 as a comparison between reference architecture and system architecture functions. For example, the function "produce product portfolio" is defined as a required function of the system architecture, which can be assigned to the reference architecture function "execute the production - produce product". The mapping then determines which logical reference architecture elements associated with the reference architecture functions can be reused and which contents need to be created from scratch. The executing stakeholder is responsible for the assessment of borderline cases (shown by dashed line), such as whether the system function "transport material to and from the production system" is covered by the reference architecture function "execute the production - produce product". In the second step, the reference architecture is adapted based on the insides derived in step one. For this purpose, all elements of the logical reference architecture that are linked to reference architecture functions, which were assigned to system functions, are obtained. For example, the reference function "execute the production - produce product" is related to the logical reference architecture element "production CSG", which is therefore considered for reuse within the system of interest. The remaining elements of the logical reference architecture that have not been considered are neglected.

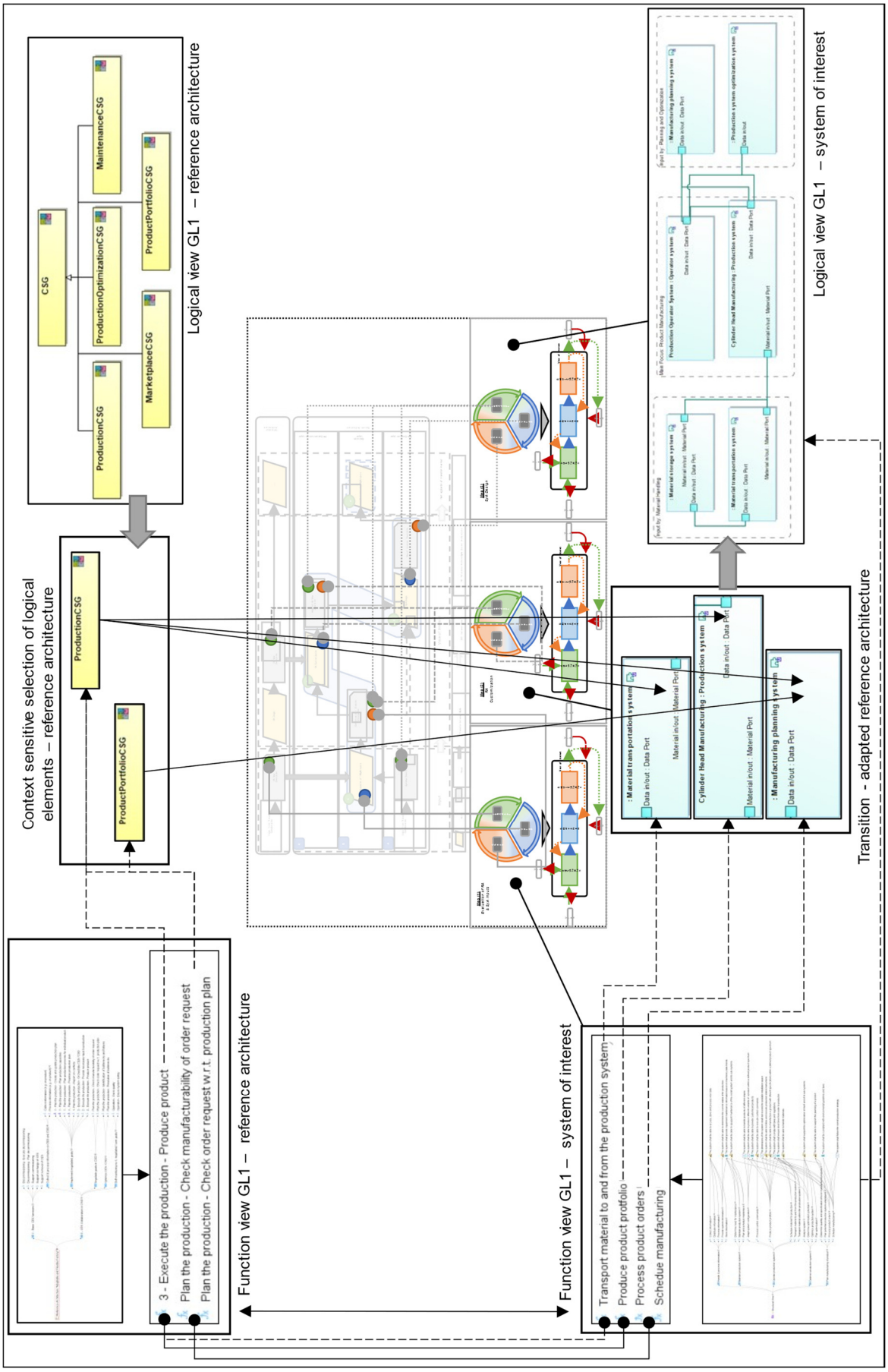


Figure 87: Logical view - exemplary transition and definition of system of interest on GL1

To complete the second step of the transition, the obtained logical reference architecture elements are adapted to the specific system and the considered system functions, if necessary. For example, based on the consideration of the system function "produce product portfolio" the logical reference architecture element "production CSG" is transformed into the logical system element "cylinder head manufacturing production system". After all elements have been adapted, the second definition step is completed. The transition is concluded by the third step, in which the existing transferred logical elements are supplemented by the logical elements that do not yet exist and the latter are set in relation to each other.

Based on the procedure described, the logical architecture of the system of interest on granularity layer one is derived within the logical view. The contents of the logical view are shown in Figure 88, for example, logical elements like the “manufacturing planning system” or the “cylinder head manufacturing production system”.

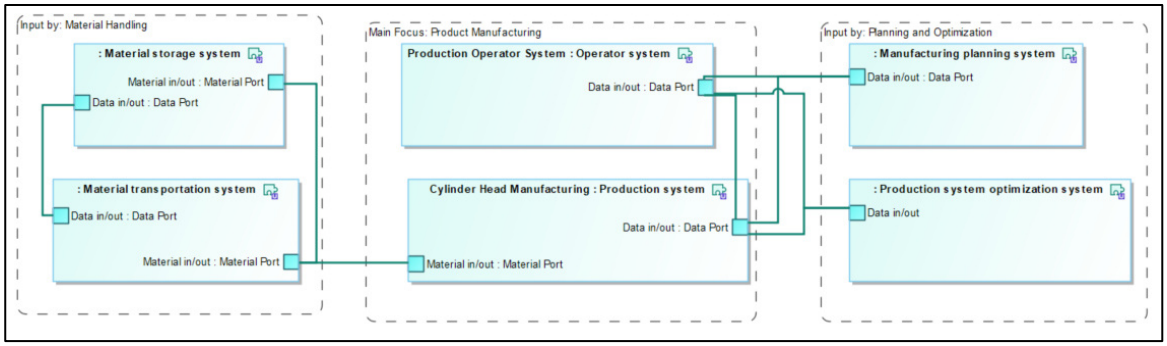


Figure 88: Logical view - logical elements of system of interest on GL1

Using the same procedure and based on the created logical view contents, the technical view on granularity layer one could be derived in a similar fashion. In this specific application scenario, the utilized reference architecture did not provide a technical reference architecture, which is why the technical view of the system of interest was created from scratch in its entirety using the architecture framework concept. The contents of the technical view of the system of interest on GL1 are shown in Figure 89.

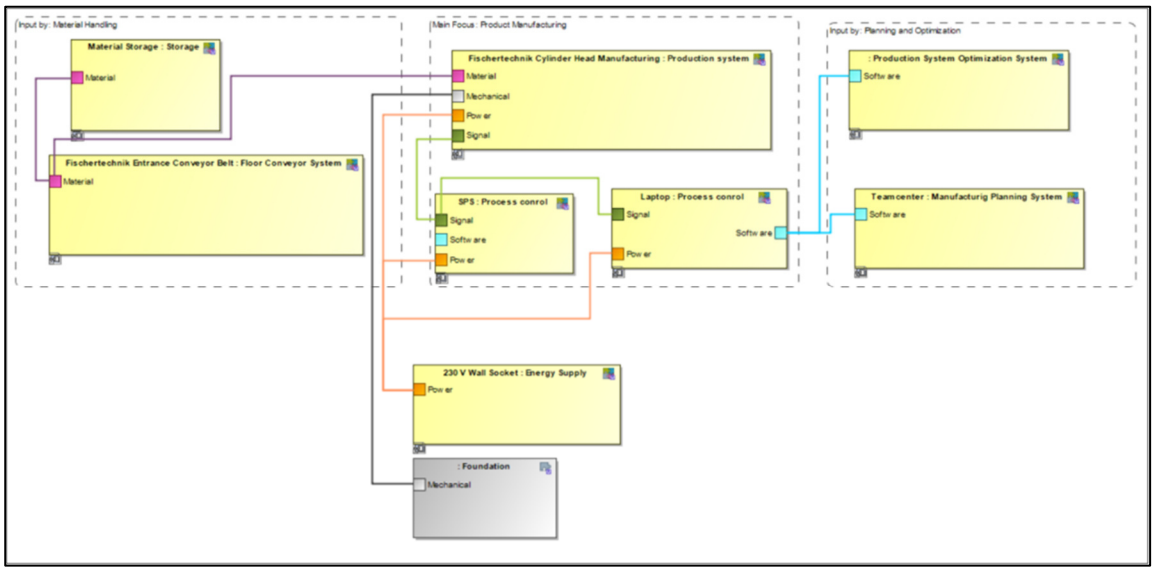


Figure 89: Technical view - technical solution of system of interest (cylinder head manufacturing) on GL1

Taking into account the defined characteristics of the architecture framework concept, the entire system of interest, namely the cylinder head manufacturing system, can be described step by step.

As an example for other views on a different granularity layer, the technical view of the system of interest on granularity layer two is shown in Figure 90. It can be seen in direct comparison of Figure 90 and Figure 89, that the technical solution is detailed from granularity layer one to granularity layer two. For example, the technical solution “Fischertechnik Cylinder Head Manufacturing : Production System” on GL1 is detailed into the technical solutions like “Mill Contour : Production System” and “Grind/Finish Cylinder Head : Production System” on GL2.

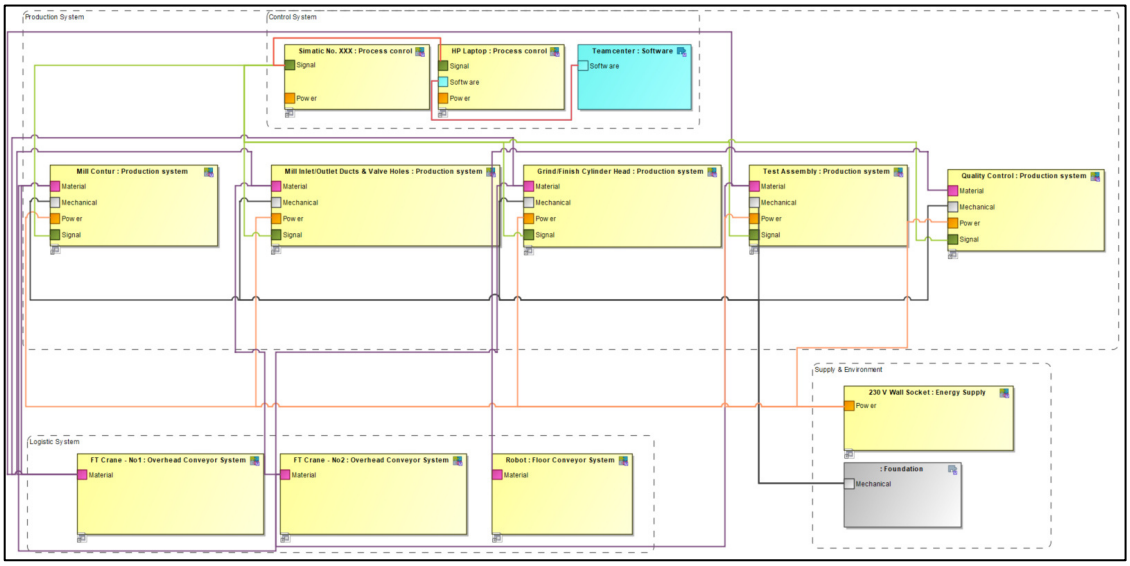


Figure 90: Technical view - technical solution of system of interest on GL2

Concluding, Figure 91 summarizes the exemplary content of granularity layer one of each viewpoint considered for the evaluation of the architecture framework concept based on the utilized cylinder head manufacturing system and shows the transition method applied.

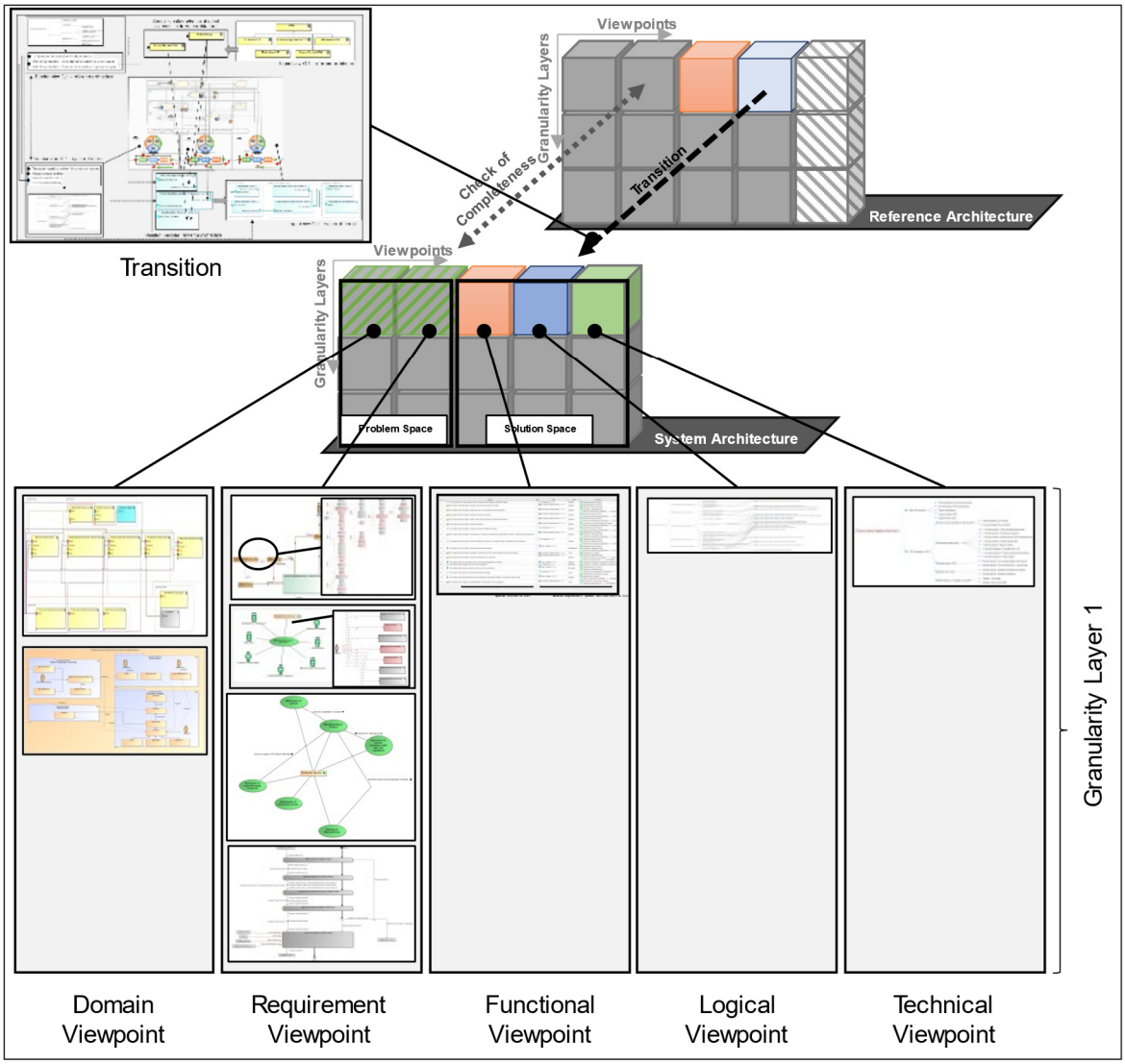


Figure 91: Cylinder head manufacturing system views of GL1 resulting from application of architecture framework concept

7.4 Results of Prototypical Application

The above-described application of the developed architecture framework concept in the form of a DSL for the specification of a system architecture description based on the use of a reference architecture description has produced different results. In the following, these are divided into the four groups labeled "reference architecture",

"system architecture", “domain-specific language, modeling tool, and involved stakeholders", and “architecture framework concept”. This subdivision serves to separate the individual results from each other better and to be able to better represent possible influences of the first three groups on the main result "architecture framework concept". The four groups and possible influences on each other are shown in Figure 92.

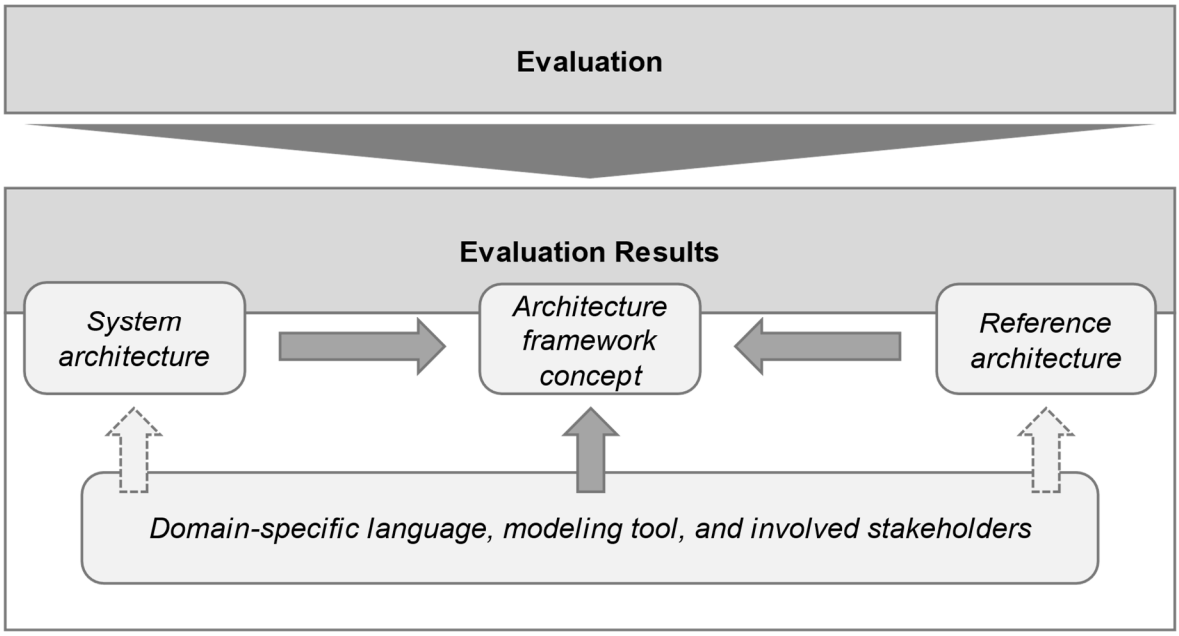


Figure 92: Structuring of evaluation of architecture framework concept application results

In the following, the findings regarding the reference architecture used, the system architecture created, and the modeling tool applied, which are not a part of the developed architecture framework concept, are considered first. If these results allow to draw potential conclusions about the architecture framework concept or future research potentials, these outcomes are also described. To sum up, the evaluation results directly related to the introduced architecture framework concept are presented.

7.4.1 Evaluation Results regarding Reference Architecture

Since an existing reference architecture was used, which corresponded to the assumptions made and the domain of interest, certain parameters such as level of abstraction, layers of granularity, and available content were given. In general, it can be stated that the utilized reference architecture provided a well-defined but, in the context of the specific production system application example, an abstract set of content. This results in a large difference between the abstraction levels, leading to

the fact that nearly all elements could be reused, but had to be adapted and supplemented. Therefore, it can be assumed, that a smaller difference between abstraction levels of reference architecture and system of interest might lead to less content reused absolutely, but it is more likely that the contents do not have to be adapted in such a way required by a large difference between abstraction levels. This assumption was made during the application of the reference architecture and might also require a concretization of the refence architecture for example by applying the transition method of the architecture framework concept (applicability needs to be evaluated in further research). Also, it became apparent that in addition to the above-mentioned assumptions with respect to the abstraction difference, an offering of several potential solution approaches (variants) within the refence architecture would have been beneficial (for example, predefined logical components for specific systems would restrict the total to be considered sphere of input and therefore reduce efforts). Furthermore, the application showed, that especially when considering architecture descriptions with larger abstraction difference, that the architect and their interpretation of the content has a strong influence on the final result. This circumstance can also be further positively or negatively influenced by the explicitness and preciseness of the individual reference architecture contents.

7.4.2 Evaluation Results regarding System Architecture

The main conclusion is that a system architecture description was successfully created that reflects the given structure, element types, and relationships that were predefined in the developed architectural framework concept. Regarding the result itself, comments should be made at this point in terms of quality and form. It should be noted that the quality of the contents of the architecture description was not assessed directly, as this depends heavily on the stakeholder, their expertise, and the available content of the reference architecture used. Rather, it was analyzed whether the representation and processes envisioned in the architecture framework concept were reflected in the system architecture description. The obtained results of the implementation correspond to the planned extent. However, this evaluation is only valid when complying with the made assumptions and within the scope of the architecture framework concept. Further application scenarios outside that scope should be considered in the future. In order to achieve results as complete as possible and to ensure that they are covered by the architecture description, further considerations, for example in the form of additional viewpoints, would have to be predefined and included in the architecture framework concept as well as implemented in the tool/DSL.

7.4.3 Evaluation Results regarding Tool, DSL, and Stakeholders

With regard to the specified DSL, it can be stated that the implemented contents of the architecture framework concept can be used intuitively and in the intended manner and that the structure and predefined element types are comprehensively represented in the solution. Due to the fact that only predefined content can be created in the customized interface of the DSL, it is easier for the user to start a new project without the necessity to specify a general structure and an implementation concept for the content to be defined. Compared to a GPL, the use of the tool requires less knowledge for the creation of content. However, it should also be noted that despite the support provided by the tool and the individually tailored design of the DSL, a certain level of architecture expertise and experience on the stakeholder's side is still needed. Furthermore, based on the prototypical state of the implementation of the architecture framework concept in the tool it became clear, that a holistic implementation of the concept would require less experience about the tool itself and the concept. For inexperienced users this fact might strongly influence the judgment on not only the quality of the results but first and foremost on the applicability of the architecture framework concept. In general, it is assumed that with regard to the model-based implementation of the framework approach in the tool, results can be better managed, shared, and used in different areas by different stakeholders. This is supported by the possibility to create individual documentation for the predefined architecture framework concept contents, which acts as a guide and contains further information as well as possible procedures in relation to a specific element type. In addition, from an user's point of view, some features of the tool that facilitate the use of the architecture framework were helpful in deriving the system architecture description. These features include, for example, the tracing between elements and the automatic representation of element relations, as well as the possibility to create and automatically compare different versions of architecture documentation.

7.4.4 Evaluation Results regarding the Architecture Framework Concept

The main result for the application of the developed architecture framework concept is that a user who is familiar with the framework concept and the creation of architectures can successfully derive a system architecture description. Thus, in the context of the prototypical application, the main goal of the thesis has been achieved, namely, to create an approach for an architecture framework considering reference architectures for system architecture descriptions. In addition, it has been confirmed that the defined structure and the predefined element types are sufficient for the

emulation of the architecture related considerations within basic and detailed engineering.

One main takeaway was that, even though the basic contents and relationships are defined by the given element types and structure, the assessment of the individual instances and whether they are relevant in the context of the transition from reference to system architecture heavily depends on the individual assessment of the stakeholder applying the architecture framework. Even though the predefined element types already limit the amount of content relevant for a transition at a certain point in time, the room for interpretation to judge the individual instantiated content is still exceptionally large. This drawback is not crucial for the intended evaluation of the principal applicability of the concept but should be investigated as a future improvement measure. Among other things, clear rules for the formulation of content or fixed designation patterns could increase the comparability of the elements. In this context, the assumption that creativity of the stakeholders is still required in order to separate content meaningfully, for example in terms of granularity layers but also in individual views, and to relate it to one another was confirmed.

Another takeaway of the evaluation was that, especially with respect to the use of reference architectures, a backwards tracing over several views was necessary in order to be able to assess whether content could be reused in the specific system of interest or not. A main reason for this situation is from the author's point of view, that with a rising degree of completeness and detailing of the system of interest the gap between the specific system content and abstract general reference architecture contents widens, which makes a comparison increasingly difficult. This problem was already described in the context of the creation of the architecture framework, but should be further examined, for example, by deriving an architecture description for a more complex system or utilizing a different reference architecture.

Furthermore, it should be noted that the expected improvement of the derivation of specific content was confirmed, with respect to the principal procedure, the structure and the contents to be defined. However, this statement is based on the simplified assumption that both the reference architecture and the architecture framework, reflected in the system architecture, have similar structures and contents. How the applicability of the framework changes with different structures and thus decreased comparability of contents should be explored in further research.

As expected from the structure and division of the content into different viewpoints, views and granularity layers, it was found that traceability, for example from requirement on GL1 to technical solutions on GL3, can only be realized by manual

step-by-step tracing or by appropriate tracing matrices in the DSL. If required, the topic of tracing can also be further elaborated in the future.

7.4.5 Conclusion on Application Results

With reference to the evaluated architecture framework concept, it can be stated that the architecture framework concept was successfully applicable for the definition of a system architecture description based on a refence architecture description. The obtained results do represent the contents expected from the prototypical application and were created with respect to the made assumptions. Most future research potentials identified during the application/evaluation are concerned with enhanced applicability or consideration of a broader scope less limited by assumption. Also, some challenges with respect to the modeling tool and the refence architecture have been identified. Overall, the scope of this thesis to provide a concept and procedure for an architecture framework, which considers refence architecture descriptions for the definition of system architecture descriptions, was achieved. The architecture framework concept developed in this thesis thus represents a solid basis on which further concepts and research projects can be build. In addition, all results of the practical application of the proposed architectural framework concept are summarized in bullet points in Figure 93.

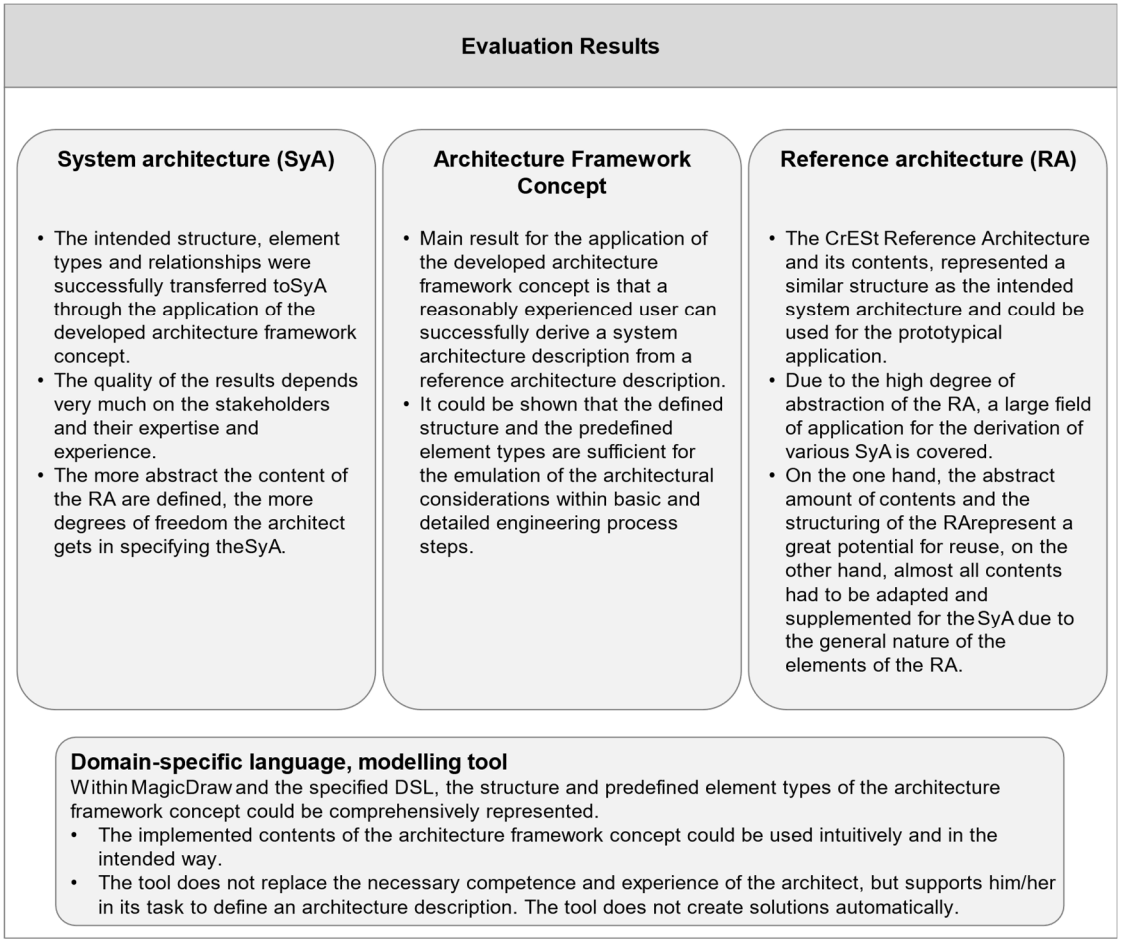


Figure 93: Summary of evaluation results

8 Summary and Outlook

The goal and result of this work is the development of an architecture framework concept as a systematic guiding methodology for the creation of system architectures based on a reference architecture used for engineering of production systems.

Sustainable business models of enterprises require flexible, supportive, cost-optimizing, and risk-minimizing approaches being used during the engineering of production systems. Architecture frameworks utilize technical, business, organizational, and product innovation knowledge as well as associated methodologies that systematically guide, structure, and support the engineering process. This development can be facilitated by the use of reference architectures that serve as templates and contain concrete solutions for a specific application area. The reference architecture and architecture framework, if properly maintained and improved throughout their life cycle, can be considered as a continuously growing knowledge and experience base that can be repeatedly applied to similar development projects.

Existing architecture frameworks are designed to define reference and system architectures separately. However, there are no established architecture frameworks that incorporate reference architectures for the definition of system architectures. Therefore, this thesis develops and proposes an architecture framework concept for creating system architecture descriptions based on reference architecture descriptions.

As a first step, the necessary definitions and concepts from the areas of engineering of production systems, architecture development, and architecture frameworks were extracted based on literature research and the state of the art. In addition, common architecture frameworks were evaluated regarding different requirements, to verify and validate the assumed research gap.

The created architecture framework concept supports the development of architecture descriptions during systems engineering with the help of a consistent basic structure, defined element types, and methodical procedures. The use of a granularity layer concept (detailing), the views contained therein (viewpoints), and associated views form the basic structure. Based on this specified structure, clear rules for the sequence of content definition, its iterative creation, relevant relationships between the elements, and change management are proposed. The concept of viewpoints comprises the domain viewpoint (domain definition), the requirement viewpoint (requirement specification), the functional viewpoint (function description), the logical viewpoint (logical realization proposals), and the technical viewpoint (technical realization), all of which are consistently applied at the different granularity layers. For

each of these viewpoints, associated views as well as element types contained therein are predefined. The developed architecture framework concept provides necessary prerequisites and specific methods for a creation and modification of relevant contents as well as for the transfer of a system architecture description from a reference architecture description.

For easier application to complex systems, better traceability, and further use of elaborated results the architecture framework concept was implemented prototypically. This was realized by following a model-based engineering approach in the modeling tool MagicDraw in the form of a specific modeling language.

Finally, the developed concept for the creation of a system architecture based on a reference architecture was successfully tested using a production system application example.

8.1 Assessment of Results Obtained in Relation to Posed Research Questions

In the following, the formulated research questions are answered with respect to the defined contents within this thesis and the obtained evaluation results.

Main Research Question - RQ1: *How to specify a system architecture description based on a predefined reference architecture utilizing an architecture framework?*

To resolve the main research question, the state of the art was considered in a first step, which confirmed the corresponding research gap, that no established architecture frameworks are considering the utilization of reference architectures for the definition of system architecture descriptions. This particularly applies to the architectural consideration within basic and detailed engineering of production systems. As stated, from the author's point of view, this is mainly caused by the combination of low availability of suitable reference architectures, poor transition process descriptions, and the higher initial investment for creation, implementing, and maintenance of reference architecture and architecture framework compared to a definition from scratch. However, over the last years organizations started to consider the topic already taken into account by research, due to, among others, the rapidly changing competitive environment and their need for securing competitiveness over time. Based on the carried-out research, relevant components for an architecture framework, which are suitable for the definition of a system architecture description

based on a reference architecture description, were defined. The focus was mainly on topics such as the relevant structure and content for the architectural considerations during engineering of a system of interest, as well as the required methods and the description of the application of this architecture framework concept (core architecture framework concept and architecture framework transition method) in the form of a guide. Structurally, different viewpoints were defined, which are detailed over several granularity layers in the form of related views. The domain, requirement, functional, logical, and technical viewpoints were considered. For an easier application also by less trained professionals, context-specific element types and models were defined for the individual views. Based on this structure, corresponding methods were developed to guide the stakeholders with respect to the topics of when and how contents can be defined in the given structure, how changes are taken into account, and how existing contents can be utilized in the creation of specific system descriptions. The guide for the application of the architecture framework concept unifies all these components and proposes resulting actions on decision questions to the architect to obtain a solution of the planned scope following the top-down approach of the elaborated architecture framework concept. To further enhance applicability and ease of use for the applying stakeholder and to carry out a prototypical application, certain parts of the developed architecture framework concept were implemented in a model-based systems engineering environment. The architecture framework concept was prototypically implemented in the form of a domain-specific language in the modeling tool MagicDraw. Based on a reference architecture developed in the research project CrESt and a production system application example the architecture framework concept was prototypically applied for the creation of a specific system architecture description utilizing a reference architecture description. Subsequently, created results and observations made during the application the concept were evaluated. The evaluation showed that the proposed architecture framework concept can be successfully used to define a system architecture description using a reference architecture in the intended scope.

Research Question RQ2: *Which structural, methodological, and content-related aspects are required within a suitable and production domain focused architecture framework for the definition of architectural descriptions?*

Based on the literature, the existing architectural frameworks, and the focus on the architectural topics of engineering, it was determined that within the core architecture framework concept a clear structure must be present in which relevant content can be defined in a logical top-down sequence. Furthermore, the classic hierarchical shape of a system suggests that this must be emulated due to a form of detailing. In addition, it was decided that appropriate methodologies for defining and modifying content must be defined. To realize the structure, five viewpoints were identified and defined. These viewpoints describe on the one hand, utilizing the domain and the requirement viewpoint, the problem space of a considered system and on the other hand, employing the functional, logical, and technical viewpoint, the solution space. These viewpoints can then be detailed using granularity layers, resulting in the typical hierarchical tree structure of the overall system. For each viewpoint and on each granularity layer, specific sub-contents were considered in the form of views, to which, if necessary, further views can be added in the future. The views include the domain and context view in the domain viewpoint, which describe the relevant environment and the interfaces between the system of interest and other elements of the environment that influence the system. The stakeholder need view, the use case view, the product view, and the requirement view describe aspects related to the requirements viewpoint. In the stakeholder need view, for example, all stakeholders relevant to the system, their tasks and interests are considered. This serves as a basis for the development of different possible use cases for the system in the use case view, and ultimately for the formulation of functional requirements, constraints, and qualities in the requirement view. In addition, when considering a production system, further use cases and requirements arise from the product to be manufactured and the associated manufacturing process. These aspects are described in the product view. Once the system of interest has been specified, the requirements are translated into appropriate solutions. This begins with the modeling of all the necessary functions that the system must provide in relation to the requirements and possible use cases. These are mapped in the functional view of the functional viewpoint. The required functionality is transformed into implementation-neutral logical elements in the logical view, which refers to the logical viewpoint. These logical elements are then transferred into technically dependent concrete solutions in the technical view of the technical viewpoint. Based on the necessary content of a production system and the current

state of the art, corresponding element types and associated relationships were defined for each view, which could be used for the specific view for the definition of relevant content when applying the architecture framework concept. Finally, two iteratively applicable methods and a guide for the application of the framework were defined. Those methods comprise a procedure for selecting the right view for the currently relevant definition step (view validity determination method) and a methodology describing the procedure for the definition of the content itself (architecture content creation method). These elements form the core architecture framework concept for the creation of an architecture description.

Research Question RQ3: *How could a possible methodological consideration of the transition between reference architecture description content for the definition of a system architecture description look like within the architecture framework? And how can the methodological components be linked to the elements of research question two so that a holistic framework approach will result in the end?*

Starting from the core architecture framework concept, a transition method was developed that relates the available content of the reference architecture to the required content of the system of interest. In a three-step methodical procedure, a statement is first made about the possible reuse and adaptation of reference architecture content in a selected view of interest, as well as about the necessity of definition from scratch. Based on this, a reference architecture description adapted to the specific system view is derived, which consists of the elements that were qualified in the first step as reusable for the system of interest. In the third and last step, the adapted reference architecture view description is complemented by the missing content for the system of interest view. The three steps are designed in such a way that the iterative architecture content creation method (three definition steps - “scope & adapt”, “develop”, and “check”) can be used. Once a view has been defined, the transition methodology can be applied step by step using the view validity determination method (“shift rules” and “preconditions”). The iterative transition method is applied until all views have been specified and a complete architecture description of the system of interest is available.

8.2 Future Research

As shown in this thesis, an architecture framework concept for the creation of system architecture descriptions based on reference architecture descriptions was successfully developed and prototypically applied. The defined architecture framework concept should be understood as a basis for further research and application tasks and serve to further advance the topic of linking reference and system architecture during definition, accompanied and guided by the architecture framework. The goal must be to reduce the effort required to create complex production systems in the medium to long term. To succeed with that goal, the most important future research questions that were derived or arose during the development and evaluation of the architecture framework concept and were not investigated are briefly summarized in the following.

With regard to the architecture framework concept and its methods, the applicability should be further evaluated by utilizing different reference architectures and specifying complex systems of various domains. Those applications should be examined, potential extension and improvement of the architecture framework concept defined and, if reasonable, implemented. In this context, it shall also be surveyed how a relaxation, change or omission of assumptions made with this thesis influences the architecture framework concept and its applicability. Furthermore, in addition to extending the form and application of the architecture framework concept, it should also be investigated how the developed concept can be integrated into more widespread holistic framework approaches, such as those pursued in enterprise architecture frameworks, in order to achieve an enhanced application of the architecture framework concept.

Regarding the reference architecture, and in contrary to the made assumptions, it should be evaluated how reference architectures, which are utilizing other structures as those within the architecture framework concept, can be considered within the current architecture framework concept and which methods might be potentially needed for the use of such reference architectures. Furthermore, it seems like, that the relation between abstraction and granularity of the reference architecture and the specific system considered within the architecture framework concept as well as the degree of predefined content within the reference architecture ("110 percent approach" or a "core approach"), have an influence on the degree of reuse and the need for adjustments. Based on those considerations, a possible influence on quality

improvement due to adapting the abstraction, granularity, and content of a reference architecture before its application should be examined in more detail. In this context, it should also be investigated if a more specific reference architecture can be derived from an available reference architecture by applying the created architecture framework concept transition method. In addition, it should be investigated how occurring changes and possible solution variants stored within the reference architecture affect the architecture framework concept and how they can be managed.

With respect to the tool, it became apparent, during the evaluation, that despite the flatter learning curve due to the tool support, additional documentation, and application guide, a relatively experienced architect is still required to create a system architecture description. Nevertheless, it was achieved, that the created environment allowed the stakeholder to focus on substantial aspects of system architecture definition. However, potential improvement points can still be found. In addition to implementing the architecture framework concept as completely as possible, for example, the development of a continuous tool chain or the automation of recurring predictable tasks can be considered.

Overall, while some potential further research approaches were identified, it was also shown in the thesis that the developed architecture framework concept can be successfully used for the specified application scenario of defining a system architecture description on the basis of as reference architecture description. Nevertheless, to further enhance the created architecture framework concept above selected research suggestions should be considered, so that the architecture framework concept can be successfully applied for the derivation of system architecture descriptions from reference architecture descriptions in the future.

Bibliography

2017

Böhm, B.; Weiß, S.; Unverdorben, S.; Schröck, S.; Vorderer, M.; Zeller, M. et al. (2017): Requirements on flexible System Architectures for collaborative embedded Systems. EC1.AP1.D1. Research Project CrESt - Document Available on Request, 2017.

2018

Böhm, B.; Zeller, M.; Vollmar, J.; Weiß, S.; Höfig, K.; Malik, V.; Unverdorben, S.; Hildebrandt, C. (2018): Challenges in the engineering of adaptable and flexible industrial factories. In: Ina Schaefer, Loek Cleophas und Michael Felderer (Hg.): Workshops at Modellierung 2018. Modellierung 2018. Braunschweig, Germany, 21.02.2018 – 23.02.2018, S. 101–110.

Unverdorben, S.; Böhm, B.; Lüder, A. (2018): Reference Architectures for Future Production Systems in the Field of Discrete Manufacturing. In: 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE). 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE). Munich, 20.08.2018 - 24.08.2018: IEEE, S. 869–874.

Maier, R.; Unverdorben, S.; Gepp, M. (2018): Efficient Implementation of Task Automation to Support Multidisciplinary Engineering of CPS. In: 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE). 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE). Munich, 20.08.2018 - 24.08.2018: IEEE, S. 1388–1393.

Weiß, S.; Böhm, B.; Unverdorben, S.; Vollmar, J. (2018): Auswirkungen zukünftiger Zusammenarbeitsszenarien auf industrielle Produktionsanlagen. Effects of future cooperation scenarios on industrial factories. In: 19. Leitkongress der Mess- und Automatisierungstechnik. Seamless Convergence of Automation & IT : Automation : Baden-Baden, 03. und 04. Juli 2018, Bd. 2330. Düsseldorf: VDI Verlag GmbH (VDI-Berichte, 2330, CD-ROM).

Vollmar, J.; Unverdorben, S.; Böhm, B.; Regnat, N. (2018): Industrial Plant Modeling Language. Domain Specific Language for Model-Based Systems Engineering. Future Architecture@Siemens 2018. Nuernberg.

Böhm, B.; Unverdorben, S.; Hohenstein, U.; Winhuysen, J.; Koo, C. H.; Carlan, C. et al. (2018): Methodology for the Design of Collaborative System Architectures (Initial Version). EC1.AP2.D1. Research Project CrESt - Document Available on Request, 2018.

2019

Daun, M.; Brings, J.; Obe, P.A.; Weiß, S.; Böhm, B.; Unverdorben, S.: Using View-Based Architecture Descriptions to Aid in Automated Runtime Planning for a Smart Factory. In: IEEE International Conference on Software Architecture Companion, ICSA Companion 2019, Hamburg, Germany, March 25-26, 2019, pages 202–209, 2019.

Unverdorben, S., Böhm, B., Lüder, A.: Industrie 4.0 – Architekturansätze und zugehörige Konzepte für konventionelle Produktionsanlagen / Industrie 4.0 – Architectural approaches and related concepts for conventional production systems. In: VDI - Verein Deutscher Ingenieure e.V. (Hg.) (2019): Automationskongress, Baden-Baden, 2019.

Caesar, B.; Grigoleit, F.; Unverdorben, S. (2019): (Self-)adaptiveness for manufacturing systems: challenges and approaches. In: SICS Software-Intensive Cyber-Physical Systems 34 (4), S. 191–200. DOI: 10.1007/s00450-019-00423-8.

Unverdorben, S.; Bohm, B.; Lüder, A. (2019 - 2019): Concept for Deriving System Architectures from Reference Architectures. In: 2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). Macao, 15.12.2019 - 18.12.2019: IEEE, S. 19–23.

Böhm, B.; Unverdorben, S.; Cârlan, C.; Voss, S.; Vollmar, J.; Weiß, S. (2019): Requirements on Tool Support for Flexible Collaborative Systems. EC1.AP3.D1.1. Research Project CrESt - Document Available on Request, 2019.

Böhm, B.; Unverdorben, S.; Koo, C. H.; Vorderer, M.; Schröck, S.; Ryashentseva, O. et al. (2019): Domain-Specific Characteristics of a Case Study. EC1.AP4.D1. Research Project CrESt - Document Available on Request, 2019.

2020

Böhm, B.; Vollmar, J.; Unverdorben, S.; Calà, A.; Wolf, S.: Ganzheitlicher modellbasierter Entwurf von Systemarchitekturen für industrielle Anlagen. Holistic Model-Based Design of System Architectures for Industrial Plants. In: Automation 2020, VDI-Bericht Nr. 2375, 2020, S. 887–898.

Vorderer, M.; Koo, C. H.; Böhm, B.; Unverdorben, S.; Albers, K.; Pollex, V. et al. (2020): Methodology for the Design of Collaborative System Architectures (Final Version). EC1.AP2.D3 v.1.1. Research Project CrESt - Document Available on Request, 2020.

C. Cârlan, B. Böhm, U. Hohenstein, S. Unverdorben, C. Kern, Y. Zhou, A. Sohr, T. Schenk, M. Allmaras, K. Albers, T. Schüle, A. Terfloth (2020): Prototypical Implementation of Tools for Flexible Collaborative Systems EC1.AP3.D1.2 & Implementation of Tools and Platforms for the Design of Flexible System Architectures EC1.AP3.D2 - Version: 1.1. Research Project CrESt - Document Available on Request, 2020.

Böhm, B.; Unverdorben, S.; Koo, C. H.; Vorderer, M.; Schröck, S.; Matruglio, O. et al. (2020): Evaluation of Application-Specific Characteristics. EC1.AP4.D2. Research Project CrESt - Document Available on Request, 2020.

2021

Böhm, B.; Cârlan, C.; Sohr, A.; Unverdorben, S.; Vollmar, J. (2021): Architectures for Flexible Collaborative Systems. In: Wolfgang Böhm, Manfred Broy, Cornel Klein, Klaus Pohl, Bernhard Rumpe und Sebastian Schröck (Hg.): Model-Based Engineering of Collaborative Embedded Systems. Extensions of the SPES Methodology. Cham: Springer International Publishing, S. 49–70.

Annex A – Overview of System Definitions

In the following, selected definitions of the term system are introduced in Table 8. The shown definitions are deemed appropriate by the author, but not considered complete. Further sources could have been added, if specific aspects should have been examined in more detail. This is not considered necessary, and the selected sources are regarded sufficient for the intended purpose of defining the term "system" with respect to this thesis. The concluding definition on the term system can be found in section 2.1.1.

Table 8: Definitions of term system

Reference	Definition
C. E. Dickerson and D. Mavris, Architecture and principles of systems engineering.	A system is a combination of interacting elements organized to realize properties, behaviors, and capabilities that achieve one or more stated purpose(s)" [45].
D. G. Firesmith, *The method framework for engineering system architectures*.	"System: a cohesive integrated set of system components (i.e., an aggregation structure) that collaborate to provide the behavior and characteristics needed to meet valid stakeholder needs and desires" [38].
IEEE - *IEEE standard glossary of software engineering terminology - IEEE Std 610.12-1990*	"[System]. A collection of components organized to accomplish a specific function or set of functions" [41].
ISO/IEC/IEEE 15288:2015- Systems and software engineering — System life cycle processes	"[Combination] of interacting elements organized to achieve one or more stated purposes Note 1 to entry: A system is sometimes considered as a product or as the services it provides. Note 2 to entry: In practice, the interpretation of its meaning is frequently clarified by the use of an associative noun, e.g., aircraft system. Alternatively, the word "system" is substituted simply by a context-dependent synonym, e.g., aircraft, though this potentially obscures a system principles perspective. Note 3 to entry: A complete system includes all of the associated equipment, facilities, material, computer programs, firmware, technical documentation, services and personnel required for operations and support to the degree

	necessary for self-sufficient use in its intended environment. system element member of a set of elements that constitute a system EXAMPLE Hardware, software, data, humans, processes (e.g., processes for providing service to users), procedures (e.g., operator instructions), facilities, materials, and naturally occurring entities or any combination. Note 1 to entry: A system element is a discrete part of a system that can be implemented to fulfill specified requirements" [37].
ISO; IEC; IEEE, *42010-2011 - ISO/IEC/IEEE Systems and software engineering: Architecture description*. Definition based on ISO/IEC/IEEE 15288	"The systems [⋯] are man - made, created and utilized to provide products or services in defined environments for the benefit of users and other stakeholders" [30]based on [37].
National Aeronautics and Space Administration, *NASA Systems Engineering Handbook*	"A "system" is the combination of elements that function together to produce the capability required to meet a need. The elements include all hardware, software, equipment, facilities, personnel, processes, and procedures needed for this purpose; that is, all things required to produce system-level results. The results include system-level qualities, properties, characteristics, functions, behavior, and performance. The value added by the system as a whole, beyond that contributed independently by the parts, is primarily created by the relationship among the parts; that is, how they are interconnected" [43].
E. Rechtin, *Systems architecting of organizations:*	"A construct or collection of different elements that together produce results not obtainable by the elements alone. The elements, or parts, can include people, hardware, software, facilities, policies, and documents; that is, all things required to produce system-level results. The results include systems-level qualities, properties, characteristics, functions, behavior, and/or performance. The value added by the system as a whole, beyond that contributed independently by the parts, is primarily created by the

	relationships among the parts; that is, how they are interconnected" [44].
U. States, Ed., *Systems engineering fundamentals*.	"Simply stated, a system is an integrated composite of people, products, and processes that provide a capability to satisfy a stated need or objective" [42].
D. D. Walden, Ed., *Systems engineering handbook: A guide for system life cycle processes and activities*	"An integrated set of elements, subsystems, or assemblies that accomplish a defined objective. These elements include products (hardware, software, firmware), processes, people, information, techniques, facilities, services, and other support elements" [26].
C. S. Wasson, *System Analysis, Design, and Development: Concepts, Principles, and Practices*.	"**System** An integrated set of interoperable elements, each with explicitly specified and bounded capabilities, working synergistically to perform value-added processing to enable a User to satisfy mission-oriented operational needs in a prescribed operating environment with a specified outcome and probability of success" [46].
VDI/VDE 3681 - Classification and evaluation of description methods in automation and control technology	"A given order of objects in the relevant context which relate to each other" [47].

Annex B – Architecture Framework Evaluation

As described in section 3.4.5, various selected architecture frameworks were evaluated with the aid of defined requirements in order to assess their support of the use of reference architectures for the definition of system architectures. The result of the evaluation is shown in Figure 26 and described in section 3.4.5.3. The individual architecture frameworks considered and the description related to the requirements are shown below in Table 9 to Table 16 in alphabetical order.

Table 9: Federal Enterprise Architectural Framework

Name	Federal Enterprise Architectural Framework (FEAF)
Objective	Improve interoperability within and between administrative and governmental structures by standardization of design and application of enterprise architectures applied within those structures.
Type	Enterprise architecture framework
Application area	Federal agency enterprise architectures
Focus on technical architecture	Low
Utilization of RA or related content	None
Transition support between RA und SyA	None
Source	[61, 132, 138, 141]

Table 10: Generalized Enterprise Reference Architecture and Methodologies (GERAM)

Name	**Generalized Enterprise Reference Architecture and Methodologies (GERAM)**
Objective	Design and maintenance of enterprises along their life cycle by providing common methods and elements relevant in both enterprise engineering and enterprise integration procedures.
Type	Enterprise architecture framework
Application area	Approach is design to be applicable in most domains – domains explicitly mentioned are, for example, industrial engineering, communication and information technology
Focus on technical architecture	Low
Utilization of RA or related content	None
Transition support between RA und SyA	None
Source	[132, 142]

Table 11: Model-Based System Architecture Process (MBSAP)

Name	**Model-Based System Architecture Process (MBSAP)**
Objective	Complexity management by providing MBSE based methods and tools
Type	System architecture framework
Application area	General approach – application areas explicitly mentioned are "mechanical, chemical, and even biological technologies and systems" [28].
Focus on technical architecture	Medium
Utilization of RA or related content	Yes (not explicitly) – only mentioned as potential reuse option and how elements could relate to reference architectures
Transition support between RA und SyA	None
Source	[28]

Table 12: The Method Framework for Engineering System Architectures (MFESA)

Name	**Method Framework for Engineering System Architectures (MFESA)**
Objective	Support architects in effective and efficient engineering of SyA
Type	System architecture framework
Application area	Systems and software engineering – not explicitly production systems
Focus on technical architecture	Medium
Utilization of RA or related content	Yes (not explicitly) – only considered as input
Transition support between RA und SyA	None
Source	[38]

Table 13: Ministry of Defense Architecture Framework (MODAF)

Name	**Ministry of Defense Architecture Framework (MODAF)**[3]
Objective	Support and enhance planning and management of defense and change activities by increasing understanding of complex topics due to capturing and representing relevant information in a comprehensive manner.
Type	Enterprise architecture framework
Application area	“Military operations and System of Systems” [61].
Focus on technical architecture	Low
Utilization of RA or related content	None
Transition support between RA und SyA	None
Source	[61, 132, 143]

Table 14: Reference Model of Open Distributed Processing (RM-ODP)

Name	Reference Model of Open Distributed Processing (RM-ODP)
Objective	Support distributed processing of interacting components by providing concepts, techniques, and rules for the specification of such systems and architectures
Type	Enterprise architecture framework
Application area	Information technology
Focus on technical architecture	Low
Utilization of RA or related content	None
Transition support between RA und SyA	None
Source	[132, 138, 144]

Table 15: The Open Group Architecture Framework (TOGAF)

Name	The Open Group Architecture Framework (TOGAF)
Objective	Standardize and reduce risk in development and implementation process of enterprise architectures, due to provision and methodological application procedure of industry best practices
Type	Enterprise architecture framework
Application area	General – development of enterprise architecture
Focus on technical architecture	Medium
Utilization of RA or related content	Yes (not explicitly) – mentioned in relation to concept of architecture repository and continuum concepts
Transition support between RA und SyA	None
Source	[61, 132, 133, 138]

Table 16: Zachman Framework

Name	Zachman Framework
Objective	Provision of tool for the development and documentation of enterprise architectures taking involved roles and objects into account by considering them from different perspectives
Type	Enterprise architecture framework
Application area	Information technology
Focus on technical architecture	Low
Utilization of RA or related content	None
Transition support between RA und SyA	None
Source	[61, 132, 138, 145]

Annex C – Content of Architecture Framework Concept Views

In connection with the use of views within the architecture framework concept, the domain view and the associated meta-model have already been presented as examples in section 5.4.2.4. The remaining view descriptions and meta-models are described below. Further information on the met models can also be found in [169]. Concluding, the relationships between the views and the associated elements are presented in Figure 102.

Domain Viewpoint

Table 17: Description of Context View

Name of View	Context View
Related Viewpoint	Domain viewpoint
Problem / Solution Space	Problem space
Goal of View	Definition of the relevant environment surrounding the system and of all elements and their relation as well as interface to the system contained therein.
Purpose of View	Specification of all influences of elements of the context and associated interfaces between system and context so that the system to be defined is clearly delimited and specified for the subsequent design steps. For all stakeholders involved in the design process, once the context and the system boundaries have been defined, it will be explicit what is and what is not part of the system. This will further facilitate communication within the project and across disciplines. The more detailed the system becomes, the more specific the problem space becomes and the narrower the solution space will be, which is the purpose of the overall design procedure.
Element Types and Model Kind of View	Element types: • Flow elements • External system • External actor

	• Environmental Effect • Packages and relationships Model kind: • Context model diagram • Context interaction scenarios
Procedure(s) within View	Based on the inputs from the domain view, within the context view, the relevant elements of the context, which have a connection and an influence on the system of interest, need to be defined. Therefore, in a first step such context elements (e.g., systems, actors) are identified. Those elements are, in a second step, set into a relation with the system of interest. By defining these relations and thus the interfaces, the system of interest is clearly separated from its context. Based on the description of the context boarder and potential influence on the system, the system itself and related requirements can be defined in the requirement viewpoint. An example of a context view is shown in Figure 81.
Relationship to other Views within Viewpoint	Domain view and context view
Relationship to other Views of other Viewpoints	Stakeholder need view, use case view, product view, and requirement view (requirement viewpoint)

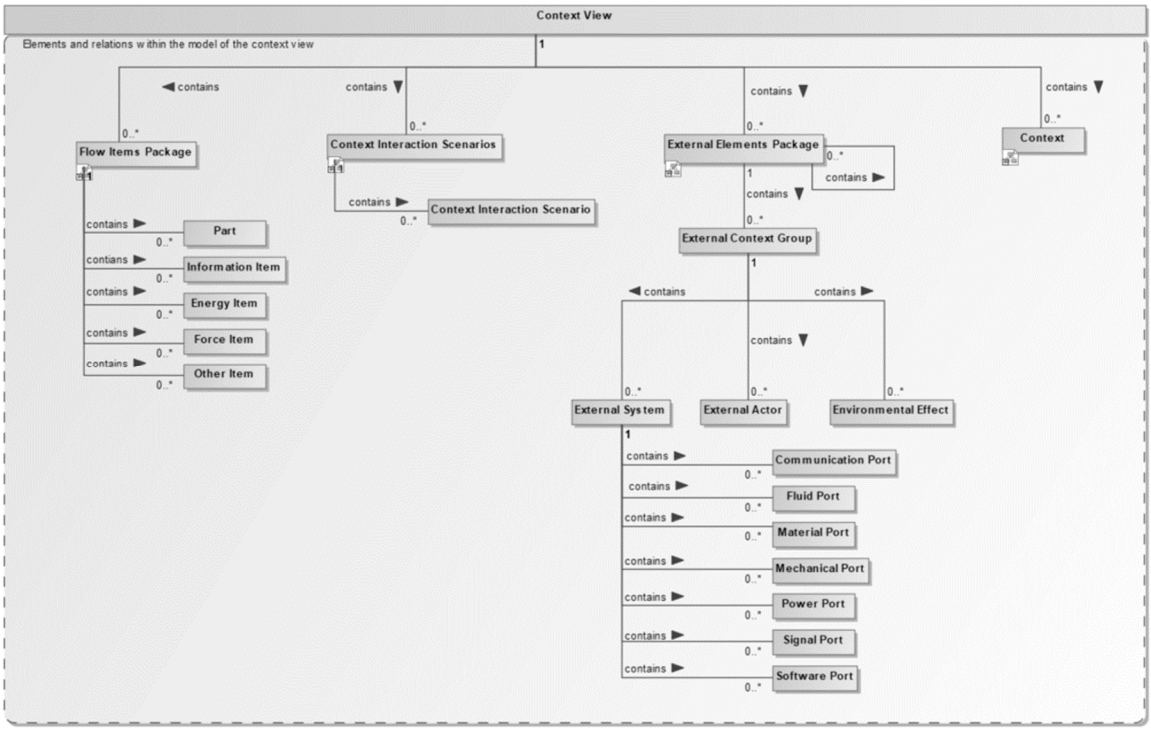


Figure 94: Meta model Context View [169]

Requirement Viewpoint

Table 18: Description of Stakeholder Need View

Name of View	Stakeholder Need View
Related Viewpoint	Requirement viewpoint
Problem / Solution Space	Problem space
Goal of View	Definition of all stakeholders related to the system, their tasks, and interest.
Purpose of View	Better understanding of the relevant stakeholders and their expectations towards the system of interest. In principle, a certain number of stakeholders have already been defined within the domain and context view. However, it is important to know and understand all internal and external stakeholders, their tasks and interest in order to define the resulting requirements for the system of interest completely and correctly. A system will only be successfully implemented if the requirements (or goals) of the stakeholders are met and an added value is created for them.
Element Types and Model Kind of View	Element types: • Stakeholder

	• Interest • Task • Packages and relationships Model kind: • Stakeholder need model diagram
Procedure(s) within View	Within the stakeholder need view all relevant stakeholders of the considered system have to be defined. Therefore, first, those stakeholders need to be identified or if already identified within the domain viewpoint specified. In order to better be able to evaluate their impact on the system, the stakeholders are further detailed by their interest and the task they perform. Based on that information, the stakeholders' expectations towards the system as well as their involvement in use cases can be specified in the following views. An example of a stakeholder need view is shown in Figure 82
Relationship to other Views within Viewpoint	Stakeholder need view, use case view, product view, and requirement view
Relationship to other Views of other Viewpoints	Domain view and context view (domain viewpoint); functional view (functional viewpoint)

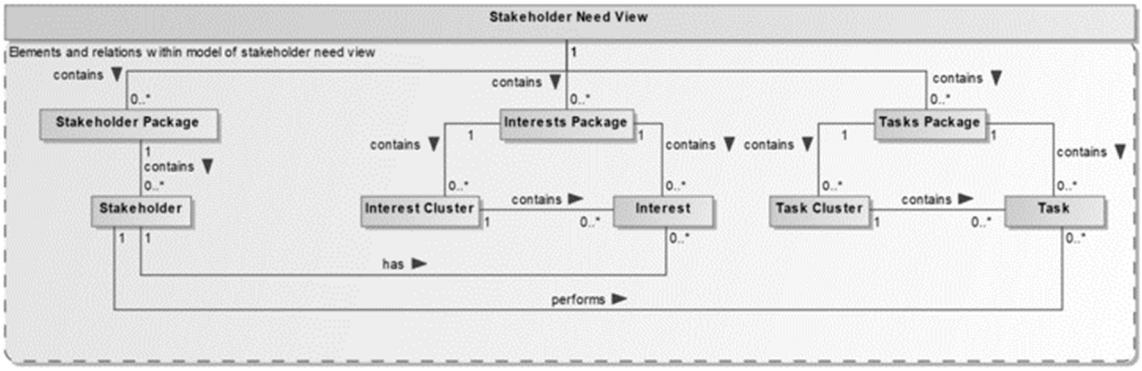


Figure 95: Meta model Stakeholder Need View [169]

Table 19: Description of Use Case View

Name of View	**Use Case View**
Related Viewpoint	Requirement viewpoint
Problem / Solution Space	Problem space
Goal of View	Definition of occurring use cases, the involved use case systems, as well as the affected use case actors in relation to the considered overall system.
Purpose of View	Further specification of the system of interest with respect to possible use cases / states in which the overall system can be in operation in order to clearly define which services/ performances the system must be able to provide and with which elements it possibly has to cope.
Element Types and Model Kind of View	Element types: • Use case system • Use case • Use case actor (stakeholder) • Packages and relationships Model kind: • Use case model diagram
Procedure(s) within View	For the definition of use cases potential application scenarios of the system have to be specified. Therefore, a specific use cases, the involved stakeholders, and systems are identified in a first step. In a second step the elements of single use case are set into a relation to each other, to be able to identify potential dependencies. In a third step, use cases are related to each other to highlight overlaps of the same elements and to derive all (mutual) dependencies on the system. An example of a use case view is shown in Figure 83.
Relationship to other Views within Viewpoint	Stakeholder need view, use case view, product view, and requirement view
Relationship to other Views of other Viewpoints	Domain view and context view (domain viewpoint); functional view (functional viewpoint)

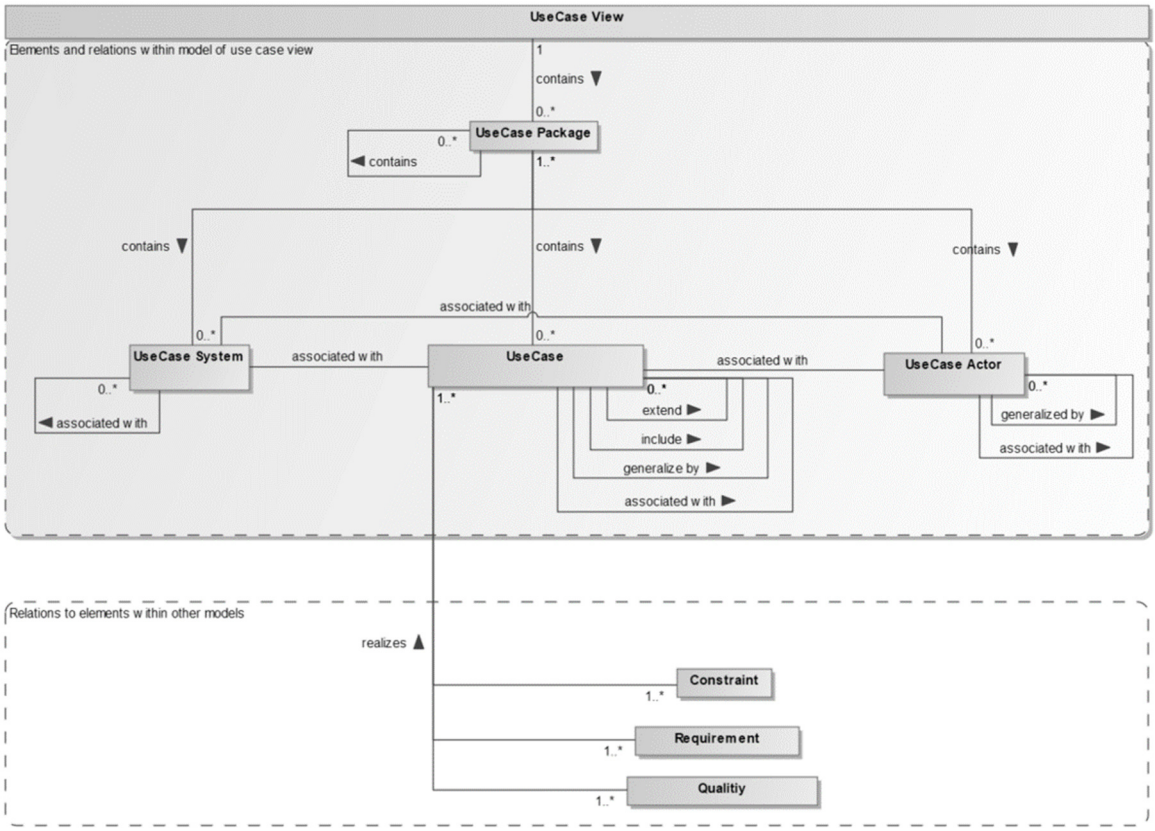


Figure 96: Meta model Use Case View [169]

Table 20: Description of Product View

Name of View	**Product View**
Related Viewpoint	Requirement Viewpoint
Problem / Solution Space	Problem Space
Goal of View	Description of the product, the manufacturing process, potential parts of the product, and required skills for manufacturing the product.
Purpose of View	Understanding and documenting the effects of a specific product and the related manufacturing process on the system of interest, so that the product can be manufactured as intended on the designed system after implementation.
Element Types and Model Kind of View	Element Types: • Part • Required skills • Production process • Attributes • Packages and relationships

	Model Kind: • Product model diagram
Procedure(s) within View	To derive relevant requirements for the system, which shall be used for manufacturing the product, the product itself, potential parts of the product, and the manufacturing process need to be defined. For this purpose, the manufacturing process is modeled step by step, which the product to be manufactured must pass through. During the step-by-step modeling, the utilized parts, required skills, and the current state of the product are described. Once the process has been completely modeled, it can be derived how the product under consideration is composed and what requirements result from the product and its manufacturing to the system of interest. An example of a product view is shown in Figure 84.
Relationship to other Views within Viewpoint	Stakeholder need view, use case view, product view, and requirement view
Relationship to other Views of other Viewpoints	Domain view and context view (domain viewpoint); functional view (functional viewpoint)

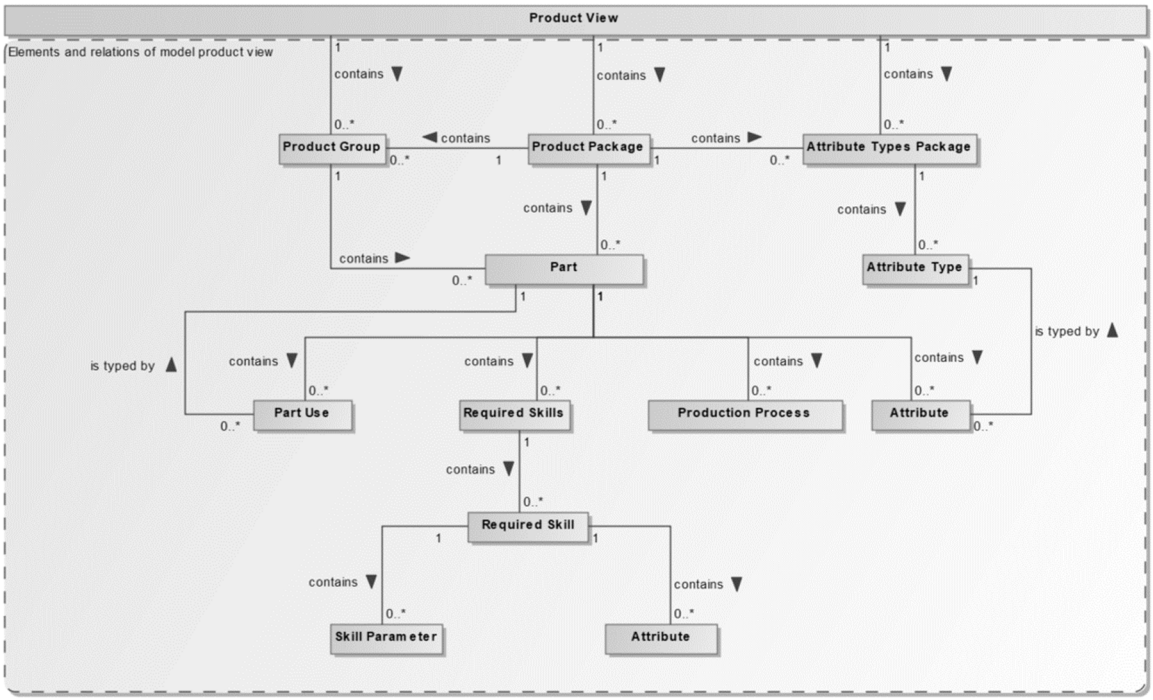


Figure 97: Meta model Product View [169]

Table 21: Description of Requirement View

Name of View	**Requirement View**
Related Viewpoint	Requirement viewpoint
Problem / Solution Space	Problem space
Goal of View	Definition of all functional requirements, qualities, and constraints necessary for the design of the system, based on the results of the upstream views.
Purpose of View	Creation of comprehensive requirements basis as a starting point for the definition of the solution space of a system of interest, in order to be able to limit the solution space accordingly and to be able to advance the development of a solution.
Element Types and Model Kind of View	Element types: • Requirement • Quality • Constraints • Packages and relationships Model kind: • Requirement model diagram
Procedure(s) within View	Based on the preliminary considerations of the system of interest, which were made in the domain and requirements viewpoint, functional requirements, qualities, and constraints concerning the system have to be captured. For this purpose, the possible requirements, qualities, and constraints are identified step by step in relation to elements from the previous views, documented, and related to the corresponding elements from which they result. The requirements, qualities, and constraints are identified in an iterative and creative process, usually performed in collaboration with the actual stakeholders of the system of interest. The defined requirements, qualities, and constraints are then used to continuously control architectural results during the development of the system. An example of a requirement view is shown in Figure 85.
Relationship to other Views within Viewpoint	Stakeholder need view, use case view, product view, and requirement view

Relationship to other Views of other Viewpoints	Domain view and context view (domain viewpoint); functional view (functional viewpoint)

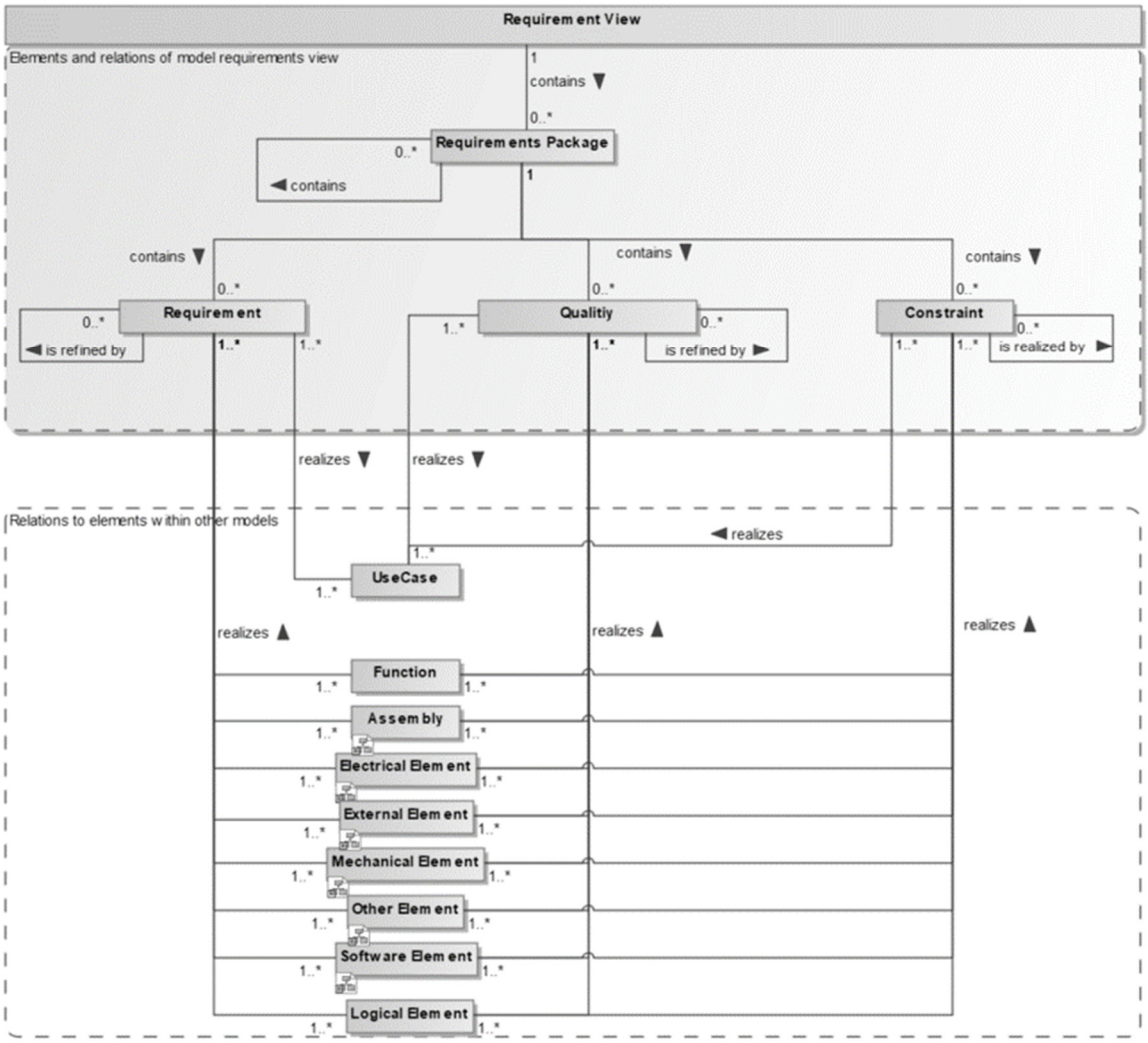


Figure 98: Meta model Requirement View [169]

Functional Viewpoint

Table 22: Description of Functional View

Name of View	Functional View
Related Viewpoint	Functional viewpoint
Problem / Solution Space	Solution space
Goal of View	Definition and documentation of all functional solutions in relation to the defined system requirements.
Purpose of View	Transition from problem to solution space and creation of a first implementation-independent solution in the form of a functional architecture, in order to understand the functional scope of the system and as a basis for the development of different logical and technical solution approaches.
Element Types and Model Kind of View	Element types: • Function • Input and output • Packages and relationships Model kind: • Function model diagram
Procedure(s) within View	Using the defined requirements for the system of interest, the architect must first examine which requirements can be expressed in the form of functions and which requirements will come into effect at a later stage of the system development, for example, during the technical design. After these basic considerations have been carried out by the architect, he/she must define appropriate functions for the first group. These must then be related both to each other and to the requirements. Finally, it must be checked whether all the necessary functions have been defined. An example of a functional view is shown in Figure 86.
Relationship to other Views within Viewpoint	Functional view
Relationship to other Views of other Viewpoints	Requirement view (requirement viewpoint), logical view (logical viewpoint), and technical view (technical viewpoint)

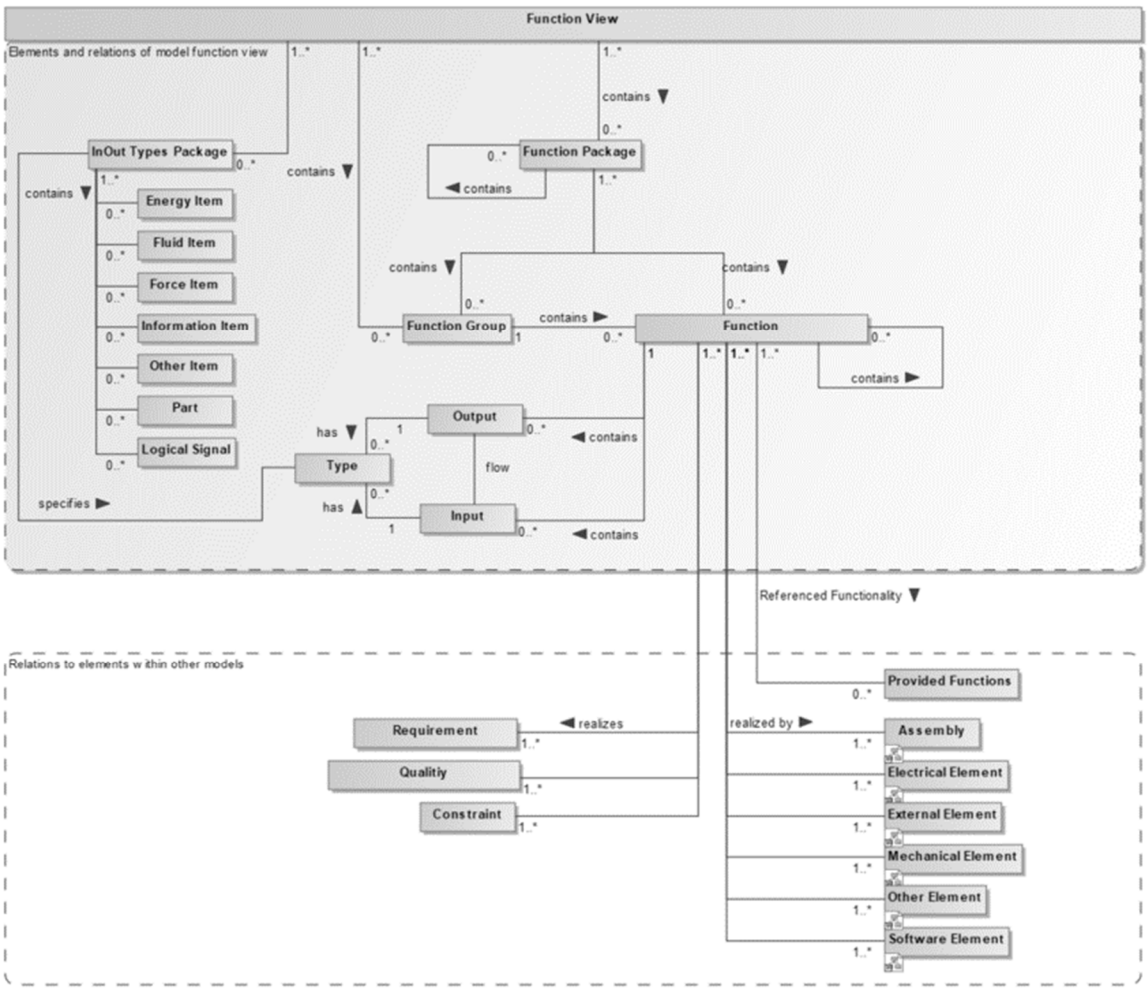


Figure 99: Meta model Functional View [169]

Logical Viewpoint

Table 23: Description of Logical View

Name of View	Logical View
Related Viewpoint	Logical viewpoint
Problem / Solution Space	Solution space
Goal of View	Definition of implementation independent logical solution elements.
Purpose of View	Transfer of the functional solution into a structure close to implementation (logical architecture), as an intermediate step to detail the final technical solution.
Element Types and Model Kind of View	Element types: • Logical system • Logical element • Provided skills • Packages and relationships Model kind: • Logical model diagram
Procedure(s) within View	Taking into account both the defined functions and the requirements, the architect must cast the system into a logical structure and define the associated logical elements that make up the system of interest. For this purpose, the architect must define how the system will be logically clustered or divided and define which functions will be performed by which logical elements. In addition, the requirements must be taken into account, which may already affect the intended structure and roles of the different logical elements. The defined logical elements must also be related to each other, taking into account both the inputs from the requirement viewpoint and from the functional viewpoint. An example of a logical view is shown in Figure 88.
Relationship to other Views within Viewpoint	Logical view
Relationship to other Views of other Viewpoints	Requirement view (requirement viewpoint), functional view (functional viewpoint), and technical view (technical viewpoint)

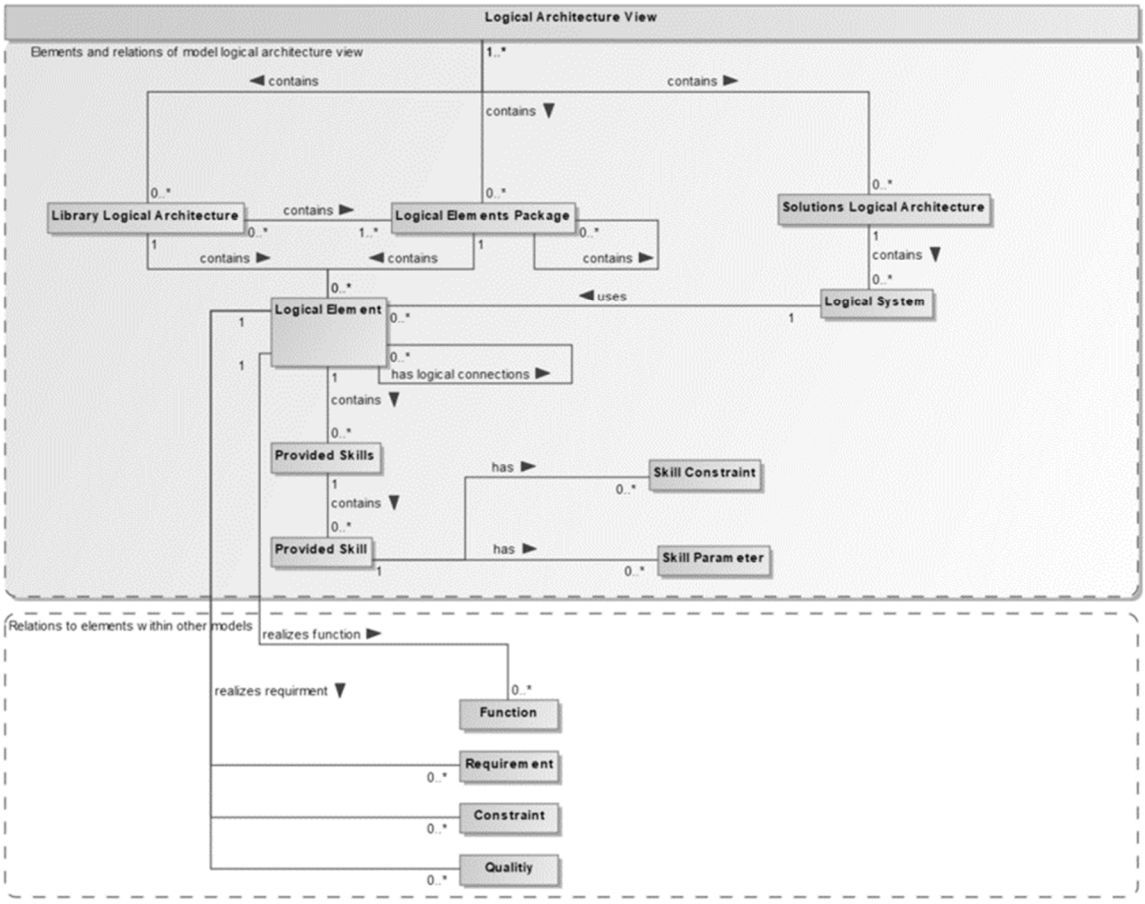


Figure 100: Meta model Logical View [169]

Technical Viewpoint

Table 24: Description of Technical View

Name of View	Technical View
Related Viewpoint	Technical Viewpoint
Problem / Solution Space	Solution Space
Goal of View	Definition of the technical system solution and all relevant sub-elements.
Purpose of View	Creation of a technical solution that meets the defined requirements and satisfies the objectives of the stakeholders and can be used as a basis for the detailed design and implementation of the system of interest.
Element Types and Model Kind of View	Element Types: • Assembly and different related elements (e.g., electrical, software, and mechanical element) • Ports, connectors, and connections • Packages and relationships Model Kind: • Technical model diagram
Procedure(s) within View	Based on the implementation-neutral structure of the logical view and its elements, the architect must derive a technical design of the system of interest, taking into account the mandatory functionalities and requirements. For this purpose, the logical elements must be converted step by step into concrete technical solutions and interconnected among each other. In addition, in particular the contents of the requirements viewpoint must be taken into account, which have an impact on for example the shape or performance of the technical solution. After the technical solution has been defined, it must be put against the requirements and checked if they are fulfilled. An example of a technical view is shown in Figure 89 and Figure 90.
Relationship to other Views within Viewpoint	Technical view
Relationship to other Views of other Viewpoints	Requirement view (requirement viewpoint), functional view (functional viewpoint), and logical view (logical viewpoint)

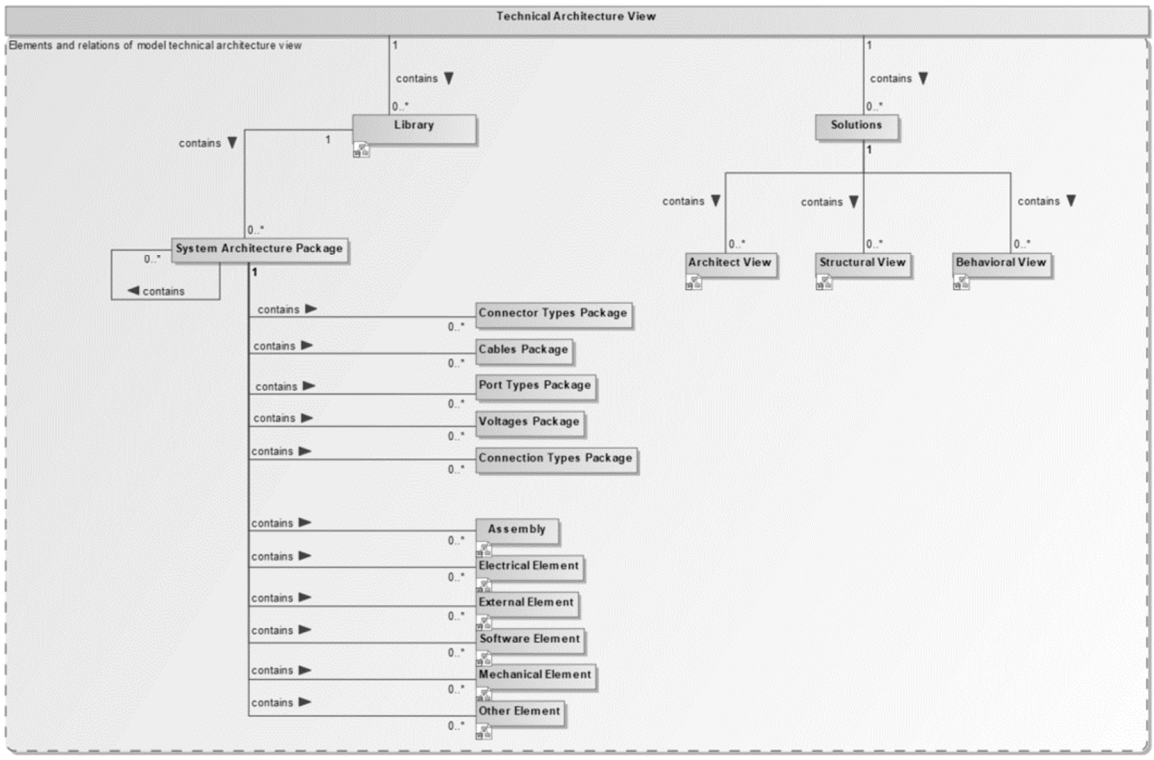


Figure 101: Meta model Technical View [169]

Overview of relationships between view and related elements

Figure 102 shows the main interrelationships of the most important elements of the different viewpoints considered during the creation of the architectural description.

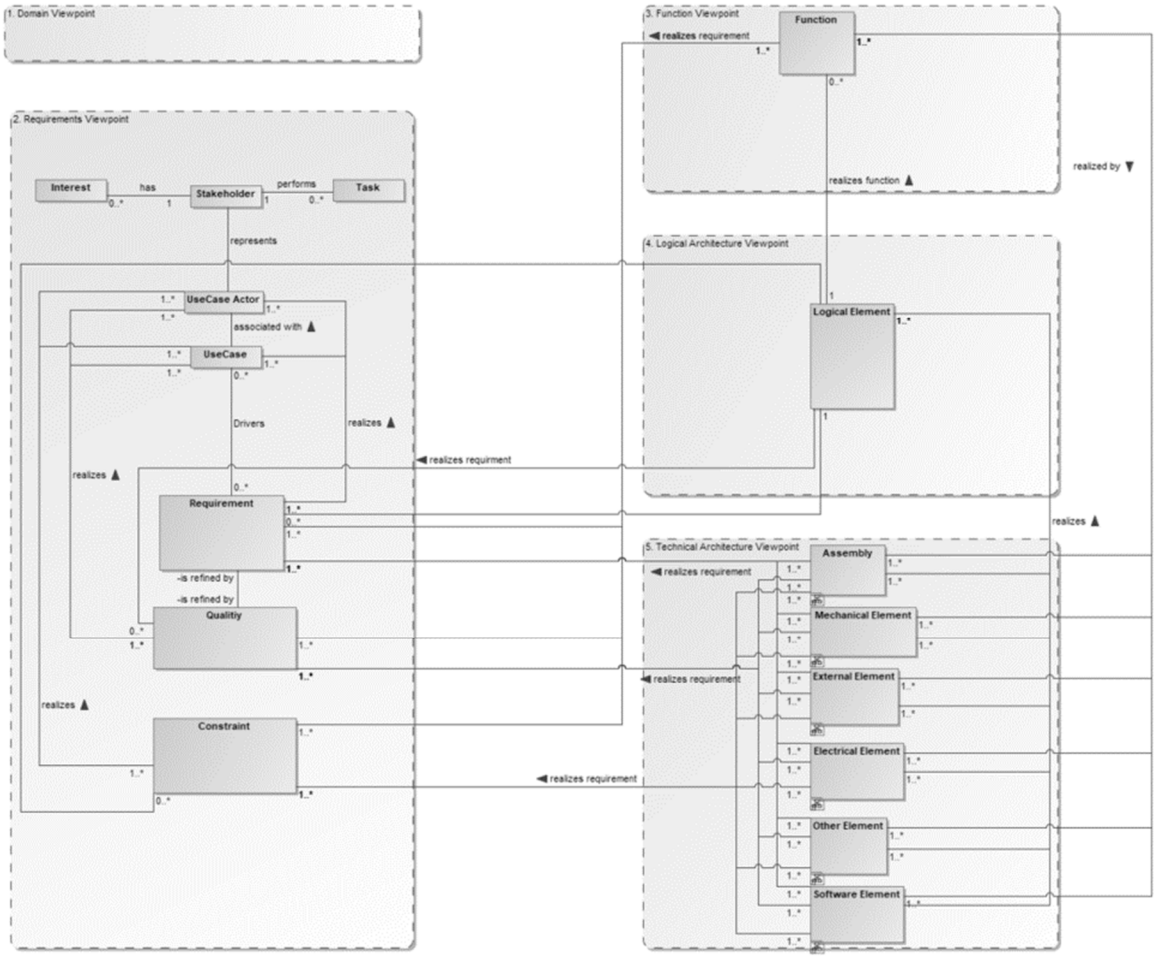


Figure 102: Overview of relationships between view and related elements

References

[1] *Systems Engineering: Historic and Future Challenges - SEBoK*. SEBoK Editorial Board. 2020. The Guide to the Systems Engineering Body of Knowledge (SEBoK), v. 2.2, R.J. Cloutier (Editor in Chief). Hoboken, NJ: The Trustees of the Stevens Institute of Technology. Accessed [DATE]. www.sebokwiki.org. BKCASE is managed and maintained by the Stevens Institute of Technology Systems Engineering Research Center, the International Council on Systems Engineering, and the Institute of Electrical and Electronics Engineers Computer Society.

[2] J. Holt, "Systems Engineering," Z-Guides - Z0 - Issue 1.1, Mar. 2020. [Online]. Available: https://incoseuk.org/Documents/zGuides/Z0_Systems_Engineering.pdf

[3] B. Beihoff *et al.,* "A World in Motion – Systems Engineering Vision 2025," 2014. [Online]. Available: https://www.incose.org/docs/default-source/aboutse/se-vision-2025.pdf

[4] M. Wolf, M. Kleindienst, C. Ramsauer, C. Zierler, and E. Winter, "Current and Future Industrial Challenges: Demographic Change and Measures for Elderly Workers in Industry 4.0," *Annals of Faculty Engineering Hunedoara – International Journal of Engineering*, no. 16, pp. 67–76, Feb. 2018.

[5] S. Biffl, A. Lüder, and D. Gerhard, *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*. Cham: Springer International Publishing, 2017.

[6] Federal Ministry of Education and Research, "Zukunftsbild „Industrie 4.0"," 2013. Accessed: Feb. 23 2018. [Online]. Available: https://www.bmbf.de/pub/Zukunftsbild_Industrie_4.0.pdf

[7] M. Maurmaier, K. Dencovski, and E. Schmitz, "Engineering Challenges - Evaluierungskonzept für Engineering-Werkzeuge: Klassifikation von Werkzeugkonzepten am Beispiel von Comos®," *ATP Magazin*, no. 1, 2008.

[8] M. Maurmaier, "Engineering Challenges & Service Challenges: Classification concepts based on challenges in industrial solution business," Stuttgart, March 16th 2007.

[9] B. Böhm *et al.,* "Challenges in the engineering of adaptable and flexible industrial factories," in *Workshops at Modellierung 2018*, Braunschweig, Germany, 2018, pp. 101–110.

[10] K. Lauenroth, F. Schreiber, and F. Schreiber, *Requirements Engineering in der Fertigungsindustrie: Kostenreduzierung und Erhöhung der Kundenzufriedenheit durch strukturierte Vorgehensmodelle,* 1st ed. Berlin: Beuth, 2016.

[11] A. Roth, *Einführung und Umsetzung von Industrie 4.0*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016.

[12] BITKOM e.V.Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V., VDMA e.V.Verband Deutscher Maschinen- und Anlagenbau e.V., and ZVEI e.V.Zentralverband Elektrotechnik- und Elektronikindustrie e.V., "Umsetzungsstrategie Industrie 4.0: Ergebnisbericht der Plattform Industrie 4.0," Apr. 2015. [Online]. Available: https://www.bitkom.org/sites/default/files/file/import/150410-Umsetzungsstrategie-0.pdf

[13] R. Mehr and A. Lüder, "Managing Complexity Within the Engineering of Product and Production Systems," in *Security and Quality in Cyber-Physical Systems Engineering*, S. Biffl, M. Eckhart, A. Lüder, and E. Weippl, Eds., Cham: Springer International Publishing, 2019, pp. 57–79.

[14] T. Tolio, *Design of Flexible Production Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[15] A. Kossiakoff, S. J. Seymour, D. A. Flanigan, and S. M. Biemer, *Systems engineering: Principles and practice*. Hoboken, NJ: John Wiley & Sons, Inc, 2020.

[16] J. Gausmeier, R. Dumitrescu, D. Steffen, A. Czaja, O. Wiederkehr, and C. Tschirner, "Systems Engineering in industrial practice," Paderborn, 2015. Accessed: 28.08.19. [Online]. Available: https://www.iem.fraunhofer.de/content/dam/iem/de/documents/Studie%20Systems%20Engineering_englisch.pdf

[17] J. Reichart, "Methodische Identifikation von Challenges im Engineering von industriellen Lösungen," Diplomarbeit, Institut für Automatisierungs- und Softwaretechnik, Universität Stuttgart, Stuttgart, 2007.

[18] A. Farncombe, "Enabling Systems Engineering: How to make SE successful in your organisation," Z-Guides - Z2 - Issue 2.0, Mar. 2009. [Online]. Available: https://incoseuk.org/Documents/zGuides/Z2_Enabling_SE.pdf

[19] A. Calà, J. Vollmar, and T. Schäffler, "Engineering in an International Context: Risks and Challenges," in *Security and Quality in Cyber-Physical Systems Engineering*, S. Biffl, M. Eckhart, A. Lüder, and E. Weippl, Eds., Cham: Springer International Publishing, 2019, pp. 33–56.

[20] U. Löwen, "Workshop: Technical Challenges in Industrial Automation Business," Erlangen, Germany, March 15th - 16th, 2007.

[21] P. Winzer, *Generic Systems Engineering: Ein methodischer Ansatz zur Komplexitätsbewältigung,* 2nd ed. Berlin, Heidelberg: Springer Vieweg, 2016.

[22] J. Gausemeier, T. Gaukstern, and C. Tschirner, “Systems Engineering Management Based on a Discipline-Spanning System Model,” *Procedia Computer Science*, vol. 16, pp. 303–312, 2013, doi: 10.1016/j.procs.2013.01.032.

[23] Deutsches Zentrum für Luft- und Raumfahrt, *Industrie 4.0.* [Online]. Available: https://www.softwaresysteme.pt-dlr.de/de/industrie-4-0.php (accessed: May 10 2020).

[24] Verein Deutscher Ingenieure (VDI), “VDI 5201: Wandlungsfähigkeit Beschreibung und Messung der Wandlungsfähigkeit produzierender Unternehmen Beispiel Medizintechnik - Adaptability Description and measurement of the adaptability of manufacturing companies Medical device industry,” 2017.

[25] Verein Deutscher Ingenieure (VDI), “VDI 2206: Design methodology for mechatronic systems,” Jun. 2004.

[26] D. D. Walden, G. J. Roedler, K. Forsberg, R. D. Hamelin, and T. M. Shortell, Eds., *Systems engineering handbook: A guide for system life cycle processes and activities,* 4th ed. Hoboken NJ u.a.: Wiley, 2015.

[27] M. Wilkinson and H. Woodcock, “System Architecture: What is system architecture?,” Z-Guides - Z8 - Issue 1.0, Oct. 2010. [Online]. Available: https://incoseuk.org/Documents/zGuides/Z8_System_Architecture.pdf

[28] J. M. Borky and T. H. Bradley, *Effective Model-Based Systems Engineering*. Cham: Springer International Publishing, 2019.

[29] G. Muller and E. Hole, Eds., “Reference Architectures; Why, What and How: White Paper Resulting from Architecture Forum Meeting,” Jun. 2007. [Online]. Available: http://www.architectingforum.org/whitepapers/SAF_WhitePaper_2007_4.pdf

[30] ISO; IEC; IEEE, *42010-2011 - ISO/IEC/IEEE Systems and software engineering: Architecture description*. Geneva, s.l., New York: ISO, 2011. [Online]. Available: http://ieeexplore.ieee.org/servlet/opac?punumber=6129465

[31] The Open Group, *The TOGAF® Standard: Website - Section 3. Defintions.* [Online]. Available: https://pubs.opengroup.org/architecture/togaf9-doc/arch/chap03.html (accessed: Aug. 29 2019).

[32] W. Böhm, *CrESt- Collaborative Embedded Systems: Research project website.* [Online]. Available: https://crest.in.tum.de/ (accessed: Apr. 15 2021).

[33] K. Pohl, H. Hönninger, and R. Achatz, *Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology*. Berlin Heidelberg: Springer Berlin

Heidelberg, 2012. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-34614-9

[34] K. Pohl, M. Broy, H. Daembkes, and H. Hönninger, Eds., *Advanced model-based engineering of embedded systems: Extensions of the SPES 2020 methodology*. Chapter 1, 3, 4. Cham, Switzerland: Springer, 2016.

[35] W. Böhm, M. Broy, C. Klein, K. Pohl, B. Rumpe, and S. Schröck, Eds., *Model-Based Engineering of Collaborative Embedded Systems: Extensions of the SPES Methodology*. Cham: Springer International Publishing, 2021.

[36] B. Böhm *et al.,* "Evaluation of Application-Specific Characteristics: EC1.AP4.D2," Research Project CrESt - Document Available on Request, Federal Ministry of Education and Research, 2020.

[37] *ISO/IEC/IEEE 15288:2015- Systems and software engineering — System life cycle processes*, 15288, ISO/IEC/IEEE, May. 2015.

[38] D. G. Firesmith, *The method framework for engineering system architectures*. Boca Raton: CRC Press, 2009. [Online]. Available: http://gbv.eblib.com/patron/FullRecord.aspx?p=427041

[39] J. Towers and H. Woodcock, "What is Model Based Systems Engineering?," Z-Guides - Z9 - Issue 2.0, Feb. 2015. [Online]. Available: https://incoseuk.org/Documents/zGuides/Z9_model_based_WEB.pdf

[40] B. Böhm *et al.,* "Use Case "Adaptable and Flexible Factory": UC.AP2.D1," Research Project CrESt - Document Available on Request, Federal Ministry of Education and Research, 2017.

[41] Institute of Electrical and Electronics Engineers, "IEEE standard glossary of software engineering terminology - IEEE Std 610.12-1990," 1990.

[42] U. States, Ed., *Systems engineering fundamentals*. [Washington D.C.]: US Army Department of Defense, 2001.

[43] National Aeronautics and Space Administration, "NASA Systems Engineering Handbook," 2016. [Online]. Available: https://www.nasa.gov/sites/default/files/atoms/files/nasa_systems_engineering_handbook_0.pdf

[44] E. Rechtin, *Systems architecting of organizations: Why eagles can't swim*. [Abingdon]: [Routledge], 2000.

[45] C. E. Dickerson and D. Mavris, *Architecture and principles of systems engineering*. Boca Raton, Fla.: CRC Press, 2010.

[46] C. S. Wasson, *System Analysis, Design, and Development: Concepts, Principles, and Practices*. Hoboken: Wiley, 2006.

[47] Verein Deutscher Ingenieure (VDI) and Verband der Elektrotechnik, Elektronik, Informationstechnik (VDE), “VDI/VDE 3681 - Classification and evaluation of description methods in automation and control technology,” 2005.

[48] Harper Collins Publishers, *Defintion of Granularity.* [Online]. Available: https://www.collinsdictionary.com/de/worterbuch/englisch/granularity (accessed: Jan. 13 2021).

[49] J. Holt and S. Perry, *SysML for Systems Engineering*: Institution of Engineering and Technology, 2008.

[50] Business Dictionary, *Definition of Granularity.* [Online]. Available: http://www.businessdictionary.com/definition/granularity.html (accessed: Feb. 4 2020).

[51] D. Fairley, K. Forsberg, and R. Madachy, *System Life Cycle Process Models: Vee.* [Online]. Available: https://www.sebokwiki.org/wiki/System_Life_Cycle_Process_Models:_Vee (accessed: Dec. 9 2020).

[52] *Systems and software engineering- Life cycle management: Part1: Guidelines for life cycle management*, 24748-1:2018, ISO/IEC/IEEE, 11 2018. [Online]. Available: https://standards.iso.org/ittf/PubliclyAvailableStandards/c072896_ISO_IEC_IEEE_24748-1_2018.zip

[53] Office of the Under Secretary of Defense for Acquisition and Sustainment, *DoDI 5000.02, "Operation of the Adaptive Acquisition Framework,".* [Online]. Available: http://acqnotes.com/wp-content/uploads/2014/09/DoD-Instruction-5000.2-Operation-of-the-Adaptive-Acquisition-Framework-23-Jan-2020.pdf (accessed: Dec. 9 2020).

[54] K. Forsberg, H. Mooz, and H. Cotterman, *Visualizing project management: Models and frameworks for mastering complex systems,* 3rd ed. Hoboken, N.J: J. Wiley, 2005. [Online]. Available: http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10305479

[55] IEEE Computer Society/Software & Systems Engineering Standards Committee, “ISO/IEC/IEEE 12207, Systems and software engineering -- Software life cycle processes,” 2017.

[56] *VDI 3695 - Engineering of industrial plants - Evaluation and optimization*, VDI 3695, Association of German Engineers (VDI), Association for Electrical; Electronic & Information Technologies (VDE), 2010 - 12.

[57] R. J. Smith, *Engineering.* [Online]. Available: https://www.britannica.com/technology/engineering (accessed: Dec. 16 2020).

[58] International Council on Systems Engineering (INCOSE), *Systems Engineering - Transforming Needs to Solutions: What is Systems Engineering?* [Online]. Available: https://www.incose.org/systems-engineering (accessed: Nov. 8 2020).

[59] T. Cusk *et al.,* "What is Systems Engineering?: Creating Successful Systems," Z-Guides - Z1 - Issue 3.0, Mar. 2009. [Online]. Available: https://incoseuk.org/Documents/zGuides/Z1_What_is_SE.pdf

[60] J. R. Armstrong, "10.1.1 A Systems Approach to Process Infrastructure," *INCOSE International Symposium*, vol. 15, no. 1, pp. 1426–1436, 2005, doi: 10.1002/j.2334-5837.2005.tb00760.x.

[61] T. Weilkiens, J. G. Lamm, M. Walker, and S. Roth, *Model-based system architecture*. Hoboken, New Jersey: John Wiley & Sons Inc, 2016. [Online]. Available: http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&AN=1081341

[62] E. S. Stelzmann, "Agile Systems Engineering," 2011. [Online]. Available: https://diglib.tugraz.at/download.php?id=576a7d0f964b0&location=browse

[63] J. Denil, R. Salay, C. Paredis, and H. Vangheluwe, "Towards Agile Model-Based Systems Engineering," *Models*, 2017.

[64] L. Z. Young, J. V. Farr, and R. Valerdi, Eds., *The role of complexities in systems engineering cost estimating processes*, 2010.

[65] E. A. Lee, "Cyber Physical Systems: Design Challenges," in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, Orlando, FL, USA, May. 2008 - May. 2008, pp. 363–369.

[66] R. Anderl *et al.,* "Fortschreibung der Anwendungsszenarien der Plattform Industrie 4.0," 2016. Accessed: Apr. 28 2021. [Online]. Available: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/fortschreibung-anwendungsszenarien.html

[67] R. Anderl *et al.,* "Aspects of the Research Roadmap in Application Scenarios," Berlin, Jul. 2016. [Online]. Available: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/aspects-of-the-research-roadmap.html

[68] L. Monostori, "Cyber-physical Production Systems: Roots, Expectations and R&D Challenges," *Procedia CIRP*, vol. 17, pp. 9–13, 2014, doi: 10.1016/j.procir.2014.03.115.

[69] S. Biffl, M. Eckhart, A. Lüder, and E. Weippl, Eds., *Security and Quality in Cyber-Physical Systems Engineering*. Cham: Springer International Publishing, 2019.

[70] A. Lüder, M. Foehr, L. Hundt, M. Hoffmann, Y. Langer, and S. Frank, "Aggregation of engineering processes regarding the mechatronic approach," in *ETFA2011*, Toulouse, France, Sep. 2011 - Sep. 2011, pp. 1–8.

[71] "The Engineers' Council for Professionel Development," *Science (New York, N.Y.)*, vol. 94, no. 2446, p. 456, 1941, doi: 10.1126/science.94.2446.456.

[72] E. W. Aslaksen, *The System Concept and Its Application to Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[73] M. W. Maier and E. Rechtin, *The art of systems architecting,* 2nd ed. Boca Raton, Fla.: CRC Press, 2002.

[74] *Information technology - Open Distributed Processing - Unified Modeling Language (UML) Version 1.4.2*, 19501, International Organization for Standardization; International Electrotechnical Commission, Apr. 2005.

[75] D. Angermeier *et al.,* "V-Modell-XT: Das deutsche Referenzmodell für Systementwicklungsprojekte - Version: 2.3," 2006.

[76] J. Gausemeier and C. Plass, *Zukunftsorientierte Unternehmensgestaltung: Strategien, Geschäftsprozesse und IT-Systeme für die Produktion von morgen,* 2nd ed. München: Hanser, 2014. [Online]. Available: http://www.hanser-elibrary.com/action/showBook?doi=10.3139/9783446438422

[77] K. Gericke, B. Bender, G. Pahl, W. Beitz, J. Feldhusen, and K.-H. Grote, "Grundlagen methodischen Vorgehens in der Produktentwicklung," in *Pahl/Beitz Konstruktionslehre*, B. Bender and K. Gericke, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2021, pp. 27–55.

[78] A. J. Qureshi, K. Gericke, and L. Blessing, "Design process commonalities in trans-disciplinary design," in *DS / The Design Society*, vol. 75, *Design for harmonies: ICED 13, the 19th International Conference on Engineering Design, 19th - 22nd August 2013, Sungkyunkwan University (SKKU), Seoul, Korea*, U. Lindemann, V. Srinivasan, Y. S. Kim, S. W. Lee, J. Clarkson, and G. Cascini, Eds., Castle Cary, Somerset: Design Society, 2013, pp. 459–468. Accessed: Mar. 24 2021. [Online]. Available: https://www.designsociety.org/publication/34870/Design+process+commonalities+in+trans-disciplinary+design

[79] A. M. Maier and H. Störrle, "Whar are the Characteristics of Engineering Design Processes?," in *Vol. 1: Design Processes, DS 68-1: Proceedings of the 18th International Conference on Engineering Design (ICED 11): Impacting Society*

through Engineering Design, S. J. Culley, B. J. Hicks, T. C. McAloone, T. J. Howard, and P. J. Clarkson, Eds., 2011, pp. 188–198. Accessed: Mar. 24 2021. [Online]. Available: https://www.designsociety.org/publication/30419/WHAT+ARE+THE+CHARACTERISTICS+OF+ENGINEERING+DESIGN+PROCESSES%3F

[80] D. C. Wynn and C. M. Eckert, "Perspectives on iteration in design and development," *Res Eng Design*, vol. 28, no. 2, pp. 153–184, 2017, doi: 10.1007/s00163-016-0226-3.

[81] A. Bleck, M. Goedecke, S. A. Huss, and K. Waldschmidt, *Praktikum des modernen VLSI-Entwurfs*. Wiesbaden: Vieweg+Teubner Verlag, 1996.

[82] D. K. Hitchins, *Systems engineering: A 21st century systems methodology*. Chichester, West Sussex, England, Hoboken, NJ: John Wiley, 2007. [Online]. Available: http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10297672

[83] R. Valerdi and M. Wheaton, "ANSI/EIA 632 As a Standardized WBS for COSYSMO," in *AIAA 5th ATIO and16th Lighter-Than-Air Sys Tech. and Balloon Systems Conferences*, Arlington, Virginia, 09262005.

[84] Electronic Industries Alliance- Engineering Department, *ANSI/EIA 632 - 1998: Processes for Engineering a System*. Arlington, VA.

[85] R. Haberfellner, P. Nagel, and M. Becker, *Systems Engineering - Methodik und Praxis*. Zürich: Verlag Industrielle Organisation, 1994.

[86] A.-P. Bröhl, Ed., *Das V-Modell: Der Standard für die Softwareentwicklung mit Praxisleitfaden,* 2nd ed. München: Oldenbourg, 1995.

[87] J. Gausemeier and J. Lückel, Eds., *Entwicklungsumgebungen Mechatronik: Methoden und Werkzeuge zur Entwicklung mechatronischer Systeme*. Paderborn: HNI Heinz Nixdorf Inst, 2000.

[88] DIN Deutsches Institut für Normung e.V., "DIN 69901: Project management – Project management systems," Jan. 2009.

[89] A. Lüdecke, "Simulationsgestützte Verfahren für den Top-Down-Entwurf heterogener Systeme," Dissertation, Fakultät für Ingenieurswissenschaften, Universität Duisburg-Essen, Duisburg-Essen, 2003. Accessed: May 31 2020. [Online]. Available: https://d-nb.info/970523009/34

[90] M. Ryan, L. Wheatcraft, R. Zinni, J. Dick, and K. Baksa, "INCOSE Guide for Writing Reqs," 2017.

[91] Defense Acquisition University, "Defense Acquisition Guidebook: Chapter 3 Systems Engineering," 2020. [Online]. Available: https://www.dau.edu/guidebooks/_layouts/15/WopiFrame.aspx?sourcedoc=

/guidebooks/Shared%20Documents/Chapter%203%20Systems%20Engineering.pdf&action=default

[92] R. Haberfellner, O. de Weck, E. Fricke, and S. Vössner, *Systems Engineering*. Cham: Springer International Publishing, 2019.

[93] J. Wikander and M. Törngren, “Mechatronics as an engineering science,” in *Mechatronics '98*, J. Adolfsson, Ed., 1st ed., s.l.: Elsevier professional, 1998.

[94] W. Koch, *Eine interaktive Entwurfsplattform für mechatronische Systeme auf der Basis von Komponentensoftware*. Zugl.: Magdeburg, Univ., Fak. für Maschinenbau, Diss., 2000. Düsseldorf: VDI-Verl., 2000.

[95] I. Graessler and J. Hentze, “Enriching Mechatronic V-Model by Aspects of Systems Engineering,” in *Computational Methods in Applied Sciences*, vol. 43, *Smart Structures and Materials: Selected Papers from the 7th ECCOMAS Thematic Conference on Smart Structures and Materials*, A. L. Araujo and C. A. Mota Soares, Eds., Cham, s.l.: Springer International Publishing, 2017. Accessed: Sep. 5 2019. [Online]. Available: http://www.dem.ist.utl.pt/smart2015/files/SMART_2015_Proceedings/PDF/Papers/SMART2015_077.pdf

[96] K. Gericke, B. Bender, G. Pahl, W. Beitz, J. Feldhusen, and K.-H. Grote, “Der Produktentwicklungsprozess,” in *Pahl/Beitz Konstruktionslehre*, B. Bender and K. Gericke, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2021, pp. 57–93.

[97] D. C. Wynn and P. J. Clarkson, “Process models in design and development,” *Res Eng Design*, vol. 29, no. 2, pp. 161–202, 2018, doi: 10.1007/s00163-017-0262-7.

[98] *Wirtschaftliche Entscheidungen beim Konstruieren: Methoden und Hilfen*, 2235, Verein Deutscher Ingenieure (VDI), 1987.

[99] D. Kristian, L. Ulrich, H. Timo, A. Michael, M. Mathias, and G. Peter, “Production System’s Life Cycle-Oriented Innovation of Industrial Information Systems,” in *Factory Automation*, J. Silvestre-Blanes, Ed.: InTech, 2010.

[100] S. Maierhofer, “Einsatz von agilen Methoden in der Produkt-/Systementwicklung,” Masterarbeit, Institut für Unternehmensführung und Organisation, Technische Universität Graz, Graz, Österreich, 2010.

[101] D. Lock, *Project management,* 9th ed. Aldershot, Hampshire: Gower, 2007. [Online]. Available: http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10211408

[102] B. S. Blanchard and J. E. Blyler, *System engineering management*. Hoboken: Wiley, 2016. [Online]. Available: http://onlinelibrary.wiley.com/book/10.1002/9781119178798

[103] R. A. Noguchi, “Recommended Best Practices based on MBSE Pilot Projects,” *INCOSE International Symposium*, vol. 29, no. 1, pp. 753–770, 2019, doi: 10.1002/j.2334-5837.2019.00633.x.

[104] T. Huldt and I. Stenius, “State-of-practice survey of model-based systems engineering,” *Syst. Engin.*, vol. 22, no. 2, pp. 134–145, 2019, doi: 10.1002/sys.21466.

[105] D. Kaslow, B. Ayres, P. T. Cahill, L. Hart, and R. Yntema, “A Model-Based Systems Engineering (MBSE) Approach for Defining the Behaviors of CubeSats,” in *2017 IEEE Aerospace Conference: Yellowstone Conference Center, Big Sky, Montana, March 4-11, 2017*, Piscataway, NJ: IEEE, 2017. Accessed: 22.02.21. [Online]. Available: https://www.incose.org/docs/default-source/default-document-library/2017-ieee-aerospace-conf---a-model-based-systems-engineering-(mbse)-approach-for-defining-the-behaviors-of-cubesats.pdf?sfvrsn=ec2189c6_0

[106] O. Alt, “Modellbasiertes Systems Engineering und seine Technologien als Schlüssel für Industrie 4.0,” in *Tag des Systems Engineering*, M. Maurer and S.-O. Schulze, Eds., München: Carl Hanser Verlag GmbH & Co. KG, 2014, pp. 1–9.

[107] S. Friedenthal, R. Griego, and M. Sampson, *INCOSE Model Based Systems Engineering (MBSE) Initiative.* [Online]. Available: https://www.researchgate.net/publication/267687693 (accessed: Nov. 10 2020).

[108] INCOSE Technical Operations, *Systems Engineering Vision 2020: Version 2.03*. Seattle, WA: International Council on Systems Engineering, 2007.

[109] B. Böhm, J. Vollmar, S. Unverdorben, A. Calà, and S. Wolf, “Ganzheitlicher modellbasierter Entwurf von Systemarchitekturen für industrielle Anlagen: Holistic Model-Based Design of System Architectures for Industrial Plants,” in *Automation 2020*, pp. 887–898.

[110] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: The systems modeling language*. Waltham, MA: Elsevier/Morgan Kaufmann, 2015.

[111] *Informationsverarbeitung in der Produktentwicklung Einführung und Wirtschaftlichkeit von EDM/PDM-Systemen: Information technology in product development Introduction and economics of EDM/PDM Systems*, 2219, Verein Deutscher Ingenieure (VDI), Berlin, Nov. 2002.

[112] A. Albers and C. Zingel, “Challenges of Model-Based Systems Engineering: A Study towards Unified Term Understanding and the State of Usage of SysML,” in *Lecture Notes in Production Engineering, Smart Product Engineering*, M.

Abramovici and R. Stark, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 83–92.

[113] S. Friedenthal, “SysML: Lessons from Early Applications and Future Directions,” *INSIGHT*, vol. 12, no. 4, pp. 10–12, 2009, doi: 10.1002/inst.200912410.

[114] M. Bone and R. Cloutier, “The Current State of Model Based Systems Engineering:Results from theOMG™ SysMLRequest for Information 2009: CSER 2010 - 8th Conference on Systems Engineering Research,” Hoboken, NJ, Mar. 2010. Accessed: Mar. 4 2021. [Online]. Available: https://www.omgsysml.org/SysML_2009_RFI_Response_Summary-bone-cloutier.pdf

[115] N. Regnat, “Why SysML does often fail – and possible solution,” in *GI-Edition / Proceedings*, P-280, *Modellierung 2018: 21.02.2018 -23.02.2018 : Braunschweig*, I. Schaefer, D. Karagiannis, A. Vogelsang, D. Méndez, and C. Seidl, Eds.

[116] U. Löwen, B. Böhm, M. Davidich, J. C. Wehrstedt, A. Campetelli, and M. Gleirischer, “Modellbasierter Entwurf in der Automatisierungstechnik,” in *VDI-Berichte, Automation 2015: Benefits of change - the future of automation ; 16. Branchentreff der Mess- und Automatisierungstechnik ; Kongresshaus Baden-Baden, 11. und 12. Juni 2015*, Düsseldorf: VDI-Verl., 2015, pp. 69–80.

[117] INCOSE, *Model - based Conceptual Design Working Group (MBCD WG) Charter.* [Online]. Available: https://www.incose.org/docs/default-source/wgcharters/model-based-conceptual-design.pdf?sfvrsn=920eb2c6_6 (accessed: 03.03.21).

[118] E. Crawley and O. deWeck, *System Architecture: An Overview and Agenda.* [Online]. Available: http://web.mit.edu/deweck/www/PDF_archive/4%20Other%20Major%20Pubs/4_2_AFOSR_MIT-Crawley_deWeck.pdf (accessed: Apr. 22 2021).

[119] Oxford University Press (OUP), *Definition of architecture.* [Online]. Available: https://www.lexico.com/definition/architecture (accessed: Mar. 9 2021).

[120] E. Fricke and A. P. Schulz, “Design for changeability (DfC): Principles to enable changes in systems throughout their entire lifecycle,” *Syst. Engin.*, vol. 8, no. 4, 2005, doi: 10.1002/sys.20039.

[121] A. M. Ross, D. H. Rhodes, and D. E. Hastings, “Defining changeability: Reconciling flexibility, adaptability, scalability, modifiability, and robustness for maintaining system lifecycle value,” *Syst. Engin.*, vol. 11, no. 3, pp. 246–262, 2008, doi: 10.1002/sys.20098.

[122] H. Sillitto, *Architecting systems: Concepts, principles and practice*. [Londres]: College Publications, op. 2014.

[123] The Open Group, *Developing Architecture Views – Introduction.* [Online]. Available: http://www.opengroup.org/public/arch/p4/views/vus_intro.htm (accessed: May 30 2020).

[124] S.-W. Lin *et al.,* "The Industrial Internet of Things Volume G1: Reference Architecture," Industrial Internet Consortium (IIC) Technology Working Group, 2017. [Online]. Available: http://www.iiconsortium.org/IIC_PUB_G1_V1.80_2017-01-31.pdf

[125] S.-W. Lin *et al.,* "Industrial Internet Reference Architecture," 2015.

[126] S. Unverdorben, B. Böhm, and A. Lüder, "Reference Architectures for Future Production Systems in the Field of Discrete Manufacturing," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, Munich, 2018, pp. 869–874.

[127] A. Bassi *et al., Enabling Things to Talk: Designing IoT solutions with the IoT Architectural Reference Model*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[128] P. Adolphs *et al.,* "Referenzarchitekturmodell Industrie 4.0 (RAMI4.0): Statusreport," Accessed: Mar. 9 2018. [Online]. Available: https://www.vdi.de/fileadmin/user_upload/VDI-GMA_Statusreport_Referenzarchitekturmodell-Industrie40.pdf

[129] M. Hankel, *ZVEI - Industrie 4.0: Das Referenzarchitekturmodell Industrie 4.0 (RAMI 4.0).* [Online]. Available: https://www.zvei.org/fileadmin/user_upload/Presse_und_Medien/Publikationen/2015/april/Das_Referenzarchitekturmodell_Industrie_4.0__RAMI_4.0_/Faktenblatt-Industrie4_0-RAMI-4_0.pdf (accessed: Mar. 9 2018).

[130] *Reference Architecture Model Industrie 4.0 (RAMI4.0)*, DIN SPEC 91345:2016-04, DIN Deutsches Institut für Normung, Apr. 2016.

[131] B. Böhm, "ProductionCES of CrESt reference architecture: unpublished - internal document," 2020.

[132] D. Matthes, *Enterprise-Architecture-Frameworks-Kompendium: Über 50 Rahmenwerke für das IT-Management*. Berlin, Heidelberg, Dordrecht, London, New York: Springer, 2011.

[133] The Open Group, *The TOGAF Standard: Version 9.2,* 9th ed. Zaltbommel: Van Haren Publishing, 2018.

[134] N. Benkamoun, W. ElMaraghy, A.-L. Huyet, and K. Kouiss, “Architecture Framework for Manufacturing System Design,” *Procedia CIRP*, vol. 17, pp. 88–93, 2014, doi: 10.1016/j.procir.2014.01.101.

[135] B. Bender and K. Gericke, Eds., *Pahl/Beitz Konstruktionslehre*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2021.

[136] J. Hentze and B. Thies, *Stakeholder-Management und Nachhaltigkeits-Reporting*. Berlin: Springer Gabler, 2014. [Online]. Available: http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=842399

[137] K. Hell, “Methoden der projektübergreifenden Wiederverwendung im Anlagenentwurf: Methoden der projektübergreifenden Wiederverwendung im Anlagenentwurf - Konzeptionierung und Realisierung in der Automobilindustrie - ,” Dissertation, Institut für Arbeitswissenschaft, Fabrikautomatisierung und Fabrikbetrieb (IAF), Otto-von-Guericke-Universität Magdeburg, Magdeburg, 2018.

[138] D. Minoli, *Enterprise architecture A to Z: Frameworks, business process modeling, SOA, and infrastructure technology*. Boca Raton: CRC Press, 2008. [Online]. Available: http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10233556

[139] L. Urbaczewski and S. Mrdalj, “A COMPARISON OF ENTERPRISE ARCHITECTURE FRAMEWORKS,” *Issues in Information Systems*, Volume VII, No. 2, 2006. [Online]. Available: http://ggatz.com/images/SOA_COMPARE.pdf

[140] U.S. Department of Defense, *Department of Defense Architecture Framework Version 2.0 (DoDAF).* [Online]. Available: https://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF%20V2%20-%20Volume%202.pdf (accessed: May 3 2021).

[141] *Federal Enterprise Architecture Framework (FEAF): Version 2.* [Online]. Available: https://obamawhitehouse.archives.gov/sites/default/files/omb/assets/egov_docs/fea_v2.pdf (accessed: Nov. 20 2020).

[142] IFIP–IFAC Task Force on Architectures for Enterprise Integration, “GERAM: Generalised Enterprise Reference Architecture and Methodology: Version 1.6.3,” 1999.

[143] Ministry of Defence, *MOD Architecture Framework (MODAF).* [Online]. Available: https://www.gov.uk/guidance/mod-architecture-framework (accessed: Dec. 3 2020).

[144] *Information technology — Open Distributed Processing — Reference Model (RM-ODP)*, 10746, Joint Technical Committee ISO/IEC JTC 1, 1998.

[145] J. A. Zachman, *The Zachman Framework.* [Online]. Available: https://www.zachman.com/about-the-zachman-framework (accessed: Nov. 26 2020).

[146] B. Caesar, F. Grigoleit, and S. Unverdorben, "(Self-)adaptiveness for manufacturing systems: challenges and approaches," *SICS Software-Intensive Cyber-Physical Systems*, vol. 34, no. 4, pp. 191–200, 2019, doi: 10.1007/s00450-019-00423-8.

[147] M. Vorderer *et al.,* "Methodology for the Design of Collaborative System Architectures (Final Version): EC1.AP2.D3 v.1.1," Research Project CrESt - Document Available on Request, 2020.

[148] S. Unverdorben, B. Böhm, and A. Lüder, "Concept for Deriving System Architectures from Reference Architectures," in *2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Macao, Macao, Dec. 2019 - Dec. 2019, pp. 19–23.

[149] *2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*: IEEE, Dec. 2019 - Dec. 2019.

[150] C. Rupp, *Requirements-Engineering und -Management: Aus der Praxis von klassisch bis agil,* 6th ed. München: Hanser, 2014.

[151] A. Calà, "A novel migration approach towards decentralized automation in cyber-physical production systems," 2019.

[152] Cambridge University Press, *Defintion of the Term Abstract.* [Online]. Available: https://dictionary.cambridge.org/de/worterbuch/englisch-deutsch/abstract (accessed: Feb. 3 2021).

[153] DIN-Normenausschuss Qualitätsmanagement, Statistik und Zertifizierungsgrundlagen (NQSZ), "DIN_EN_ISO_9000: Quality management systems – Fundamentals and vocabulary (ISO 9000:2015); German and English version EN ISO 9000:2015," 2015.

[154] *Methodisches Entwerfen technischer Produkte: Systematic embodiment design of technical products*, 2223, Berlin, Jan. 2004.

[155] *Design of technical products and systems - Model of product design*, 2221, VDI.

[156] A. Sloman, "Exploring design space and niche space," *Proceedings 5th Scandinavian Conf. on AI*, May. 1995.

[157] B. Westerlund, *Design Space conceptual tool: grasping the design process*, 2005.

[158] H. Li and R. Lachmayer, “Automated Exploration of Design Solution Space Applying the Generative Design Approach,” *Proc. Int. Conf. Eng. Des.*, vol. 1, no. 1, pp. 1085–1094, 2019, doi: 10.1017/dsi.2019.114.

[159] P. T. Ward and S. J. Mellor, *Structured development for real time systems*. New York, NY: Yourdon Pr, 1985.

[160] B. Nuseibeh, “Weaving together requirements and architectures,” *IEEE Computer*, 34 (3), pp. 115–119, 2001, doi: 10.1109/2.910904.

[161] T. Berger, “Softwareproduktlinie-Entwicklung - Domain Engineering: Konzepte, Probleme und Lösungsansätze: Betrachtung im Rahmen einer Fallstudie über Entwicklung eines Portals und eines Frameworks zur Unterstützung elektronischer Prüfungsabläufe,” 2007.

[162] T. Taura, Ed., *Principia designae - pre-design, design, and post-design: Social motive for the highly advanced technological society*. Tokyo: Springer, 2015. [Online]. Available: https://www.loc.gov/catdir/enhancements/fy1617/2014942844-d.html

[163] D. Olsen, *The lean product playbook: How to innovate with minimum viable products and rapid customer feedback*. Hoboken, NJ: Wiley, 2015. [Online]. Available: http://proquest.safaribooksonline.com/9781118960875

[164] E. Evans, “Domain-– - Driven Design Reference: Definitions and Pattern Summaries,” 2015.

[165] R. Kluge and K. Schloer, *Requirements Engineering: A short RE Primer.* [Online]. Available: https://www.sophist.de/fileadmin/user_upload/Bilder_zu_Seiten/Publikationen/Wissen_for_free/RE-Broschuere_Englisch_-_Online.pdf (accessed: Dec. 7 2020).

[166] R. L. Ackoff, “Towards a System of Systems Concepts,” *Management Science*, Vol. 17, pp. 661–671, Jul. 1971. [Online]. Available: https://www.jstor.org/stable/2629308?seq=8#metadata_info_tab_contents

[167] Cambridge University Press, *Defintion of the Term Goal.* [Online]. Available: https://dictionary.cambridge.org/us/dictionary/english/goal-definition (accessed: Dec. 1 2020).

[168] Cambridge University Press, *Defintion of the Term Purpose.* [Online]. Available: https://dictionary.cambridge.org/us/dictionary/english/purpose (accessed: Dec. 1 2020).

[169] C. Carlan *et al.,* "Prototypical Implementation of Tools for Flexible Collaborative Systems EC1.AP3.D1.2 & Implementation of Tools and Platforms for the Design of Flexible System Architectures EC1.AP3.D2 - Version: 1.1," Research Project CrESt - Document Available on Request, 2020.

[170] V. Crespi, A. Galstyan, and K. Lerman, "Top-down vs bottom-up methodologies in multi-agent system design," *Auton Robot*, vol. 24, no. 3, pp. 303–313, 2008, doi: 10.1007/s10514-007-9080-5.

[171] A. Suter, S. Vorbach, and D. Wild-Weitlaner, *Die Wertschöpfungsmaschine: Prozesse und Organisation aus der Strategie ableiten,* 2nd ed. München: Hanser, 2019.

[172] M. H. Weik, "top-down definition," in *Computer Science and Communications Dictionary*, M. H. Weik, Ed., Boston, MA: Springer US, 2001, p. 1797.

[173] G. Müller-Stewens, *Top-Down-Prinzip.* [Online]. Available: https://wirtschaftslexikon.gabler.de/definition/top-down-prinzip-49846/version-273072 (accessed: Jun. 3 2020).

[174] M. H. Weik, "top-down designing," in *Computer Science and Communications Dictionary*, M. H. Weik, Ed., Boston, MA: Springer US, 2001, p. 1797.

[175] M. H. Weik, "bottom-up," in *Computer Science and Communications Dictionary*, M. H. Weik, Ed., Boston, MA: Springer US, 2001, p. 140.

[176] R. Lackes and M. Siepermann, *Bottom-up-Prinzip.* [Online]. Available: https://wirtschaftslexikon.gabler.de/definition/bottom-prinzip-27383/version-251039 (accessed: Jun. 3 2020).

[177] M. H. Weik, "bottom-up designing," in *Computer Science and Communications Dictionary*, M. H. Weik, Ed., Boston, MA: Springer US, 2001, p. 141.

[178] REFA AG, *PDCA-Zyklus.* [Online]. Available: https://refa.de/service/refa-lexikon/pdca-zyklus (accessed: 12.05.21).

[179] H.-D. Zollondz, *Grundlagen Qualitätsmanagement: Einführung in Geschichte, Begriffe, Systeme und Konzepte*: De Gruyter.

[180] W. F. Daenzer and F. Huber, *Systems Engineering – Methoden und Praxis: 8. verbesserte Auflage*. Zürich: Industrielle Organisation, 1994.

[181] K.-D. Thoben, *CAD: Sparen durch Wiederhol-Konstruktionen*. Zugl.: Bremen, Univ., Diss. u.d.T.: Thoben, Klaus-Dieter: Ausnutzung systematisch vorbereiteter Wiederholeffekte zur Senkung des Konstruktionsaufwands. Düsseldorf: VDI-Verl., 1990.

[182] Ł. Hady, *Entwicklung einer online-basierten Modulbibliothek zur Steigerung der Planungsqualität, Know-how-Sicherung und Wiederverwendung des Engineering bei der modularen Anlagenplanung*. Zugl.: Berlin, Techn. Univ., Diss., 2013. Berlin: Logos-Verl., 2013.

[183] S. Knabe, "Bewertung der Verwendbarkeit von Komponenten im Produktionssystem am Beispiel Karosseriebau.," Masterarbeit, Magdeburg, 2015.

[184] E. Bursa, R. Fusaro, and D. Ferretto, "Practical application of the Model Based Systems Engineering to the product development and introduction to its tools and language," Erlangen, Germany, Apr. 10 2019.

[185] A. Fleischmann, S. Oppl, W. Schmidt, and C. Stary, *Ganzheitliche Digitalisierung von Prozessen*. Wiesbaden: Springer Fachmedien Wiesbaden, 2018.

[186] A. Blunk and J. Fischer, "Prototyping Domain Specific Languages as Extensions of a General Purpose Language," in vol. 7744, *System Analysis and Modeling: Theory and Practice*, D. Hutchison et al., Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 72–87.

[187] L. Delligatti, *SysML distilled: A brief guide to the systems modeling language*. Upper Saddle River, NJ: Addison-Wesley, 2014. [Online]. Available: http://proquest.tech.safaribooksonline.de/9780133430356

[188] M. Fowler, *UML distilled: A brief guide to the standard object modeling language,* 3rd ed. Boston, MA: Addison-Wesley, 2010.

[189] M. Mernik, J. Heering, and A. M. Sloane, "When and how to develop domain-specific languages," *ACM Comput. Surv.*, vol. 37, no. 4, pp. 316–344, 2005, doi: 10.1145/1118890.1118892.

[190] A. van Deursen, P. Klint, and J. Visser, "Domain-Specific Languages An Annotated Bibliography*," *ACM SIGPLAN Notices*, 35 (6), pp. 26–36, Jan. 2000. [Online]. Available: https://www.researchgate.net/publication/220178552_Domain-Specific_Languages_An_Annotated_Bibliography

[191] B. Böhm *et al.,* "Methodology for the Design of Collaborative System Architectures (Initial Version): EC1.AP2.D1," Research Project CrESt - Document Available on Request, Federal Ministry of Education and Research, 2018.

[192] T. Kosar *et al.,* "Comparing general-purpose and domain-specific languages: An empirical study," *ComSIS*, vol. 7, no. 2, pp. 247–264, 2010, doi: 10.2298/CSIS1002247K.

[193] C. Lüth, "Praktische Informatik 3 (WS 2016/17): Domänenspezifische Sprachen (DSLs)," Bremen, Jan. 17 2017.

[194] D. Bruce, “What makes a good domain-specific language? Apostle, and its approach to parallel discrete event simulation,” in *First ACM SIGPLAN Workshop on Domain-Specific Languages*, Paris, France, Jan. 1997.

[195] A. van Deursen and P. Klint, “Little languages: little maintenance?,” *J. Softw. Maint: Res. Pract.*, vol. 10, no. 2, pp. 75–92, 1998, doi: 10.1002/(SICI)1096-908X(199803/04)10:2<75::AID-SMR168>3.0.CO;2-5.

[196] D. A. Ladd and J. C. Ramming, “Two application languages in software production,” Oct. 1994. [Online]. Available: https://www.researchgate.net/publication/2802835_Two_Application_Languages_in_Software_Production

[197] J. Den Haan, *DSL development: 7 recommendations for Domain Specific Language design based on Domain-Driven Design.* [Online]. Available: http://www.theenterprisearchitect.eu/blog/2009/05/06/dsl-development-7-recommendations-for-domain-specific-language-design-based-on-domain-driven-design/ (accessed: Mar. 5 2021).

[198] SparxSystems, “Enterprise Architect 12 Reviewer's Guide,” 2019. [Online]. Available: https://www.sparxsystems.de/fileadmin/user_upload/pdfs/EAReviewersGuide_EA-141-DE.pdf

[199] No Magic, *MagicDraw: Architecture Made Simple.* [Online]. Available: https://www.nomagic.com/products/magicdraw (accessed: 27.02.21).

[200] Eclipse, *Capella.* [Online]. Available: https://www.eclipse.org/capella/ (accessed: 27.02.21).

[201] Eclipse, *Papyrus.* [Online]. Available: https://www.eclipse.org/papyrus/ (accessed: 27.02.21).

[202] R. Maier, S. Unverdorben, and M. Gepp, “Efficient Implementation of Task Automation to Support Multidisciplinary Engineering of CPS,” in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, Munich, 2018, pp. 1388–1393.

[203] J. Vollmar, S. Unverdorben, B. Böhm, and N. Regnat, “Industrial Plant Modeling Language: Domain Specific Language for Model-Based Systems Engineering,” Future Architecture@Siemens 2018, Nuernberg, Sep. 2018.

[204] Rheinisch-Westfälischen Technischen Hochschule (RWTH) Aachen, *MontiCore: Language Workbench and Development Tool Framework.* [Online]. Available: https://monticore.github.io/monticore/ (accessed: Apr. 25 2021).

[205] Eclipse Foundation, *Xtext.* [Online]. Available: http://www.eclipse.org/Xtext/ (accessed: Apr. 25 2021).

[206] Eclipse Foundation, *mbeddr.* [Online]. Available: http://mbeddr.com/ (accessed: Apr. 25 2021).

[207] No Magic, *MagicDraw - Architecture Made Simple: User Manual 18.1.* [Online]. Available: https://www.nomagic.com/files/manuals/MagicDraw%20UserManual.pdf (accessed: Apr. 26 2021).

[208] B. Böhm, S. Unverdorben, C. Cârlan, S. Voss, J. Vollmar, and S. Weiß, "Requirements on Tool Support for Flexible Collaborative Systems: EC1.AP3.D1.1," Research Project CrESt - Document Available on Request, Federal Ministry of Education and Research, 2019.

[209] S. Kranz *et al.,* "MQ1.AP2.D1 / MQ1.AP2.D2 / MQ1.AP2.D3: Modeling Methods, Views and Languages," 2018.

[210] R. Kazman, M. Klein, and P. Clements, *ATAM: Method for Architecture Evaluation: CMU/SEI-2000-TR-004, ESC-TR-2000-004.* Accessed: 24.04.21. [Online]. Available: https://www.researchgate.net/publication/2814191_ATAM_Method_for_Architecture_Evaluation

Glossary

Architecting	“Process of conceiving, defining, expressing, documenting, communicating, certifying proper implementation of, maintaining and improving an architecture throughout a system’s life cycle” [30].
Architecture	Architecture is defined as “(system) fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution” [30].
Architecture Description	An architecture description can be defined as “work product used to express an architecture” [30] of a system.
Architecture Framework	An architecture framework represents ”[...] conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders” [30].
Artifact	An artifact refers to all "tangible and intangible project deliverables" [56]. Tangible artifacts refer to, for example, "engines, pumps, [or] functional modules" [56]. Intangible artifacts refer to, for example, "plans, architectures, [or] specifications" [56].
Bottom-Up Approach	The bottom-up approach is used to design systems “[...] by starting with the most basic or primitive components and proceeding to higher-level components or modules by using the lower-level tested and approved components as building blocks, until the system design is completed” [177].
Domain	A domain can be defined as a “[...] sphere of knowledge, influence or activity [⋯]” [164]. With respect to [164] the subject area to which a stakeholder applies systems engineering procedures and methods is the domain of the system of interest.
Domain-specific Language	A domain-specific language limits and focuses on a particular application domain, provides appropriate or established notations, and the right level of abstraction to view system solutions in a natural but not overly detailed way [189, 192].
Engineering	“The application of a systematic, disciplined, quantifiable approach to structures, machines, products, systems, or processes” [41].

Engineering Process An engineering process is defined "[…] as a sequence of activities of creative application of scientific principles to design or develop structures, machines, apparatus, or manufacturing processes; all as respects an intended function, economic and safe operation" [70] following [71].

Environment The environment of a system considers all elements outside the defined boundary of the system of interest. Therefore, the environment includes relevant and irrelevant elements with respect to the definition of the system of interest. Elements of the environment might be connected by relationships among each other and to the system of interest.

Function A function is an action or task provided and executed by a system with the aim of fulfilling the goal and the defined purpose for which the system has been created [82, 166].

General Purpose Language A general purpose language is a cross-domain modeling language with language constructs not restricted to one particular domain [186]. The language represents reality as well as relevant interrelationships and is utilized for implementing them in models.

Goal "The act of stating clearly what you want to achieve or what you want someone else to achieve" [167].

Granularity "The level of detail considered in a model or decision making process. The greater the granularity, the deeper the level of detail. Granularity is usually used to characterize the scale or level of detail in a set of data" [50].

Level of Abstraction The abstraction level relates systems of a particular domain/environment to each other and provides information about their level of detail in relation to other systems within the domain/environment. That is, to place them in a span between abstract and concrete system classification. Following the ISO 9000 standard [153], systems at a low level of abstraction inherit all the characteristics of higher-level concepts and contain information that distinguishes them from systems at the same level.

Life Cycle A life cycle defines an "evolution of a system, product, service, project or other human-made entity from conception through retirement" [37].

Life Cycle Model	A life cycle model is a “framework of processes and activities concerned with the life cycle that may be organized into stages, which also acts as a common reference for communication and understanding” [37].
Logical Component	A logical component represents the realization of one or more required system functions on a solution-oriented, technically implementation-free level.
Model	“A model captures a view of a physical system. Hence, it is an abstraction of the physical system with a certain purpose; for example, to describe behavioral aspects of the physical system to a certain category of stakeholders. A model contains all the model elements needed to represent a physical system completely according to the purpose of this particular model. The model elements in a model are organized into a package/subsystem hierarchy, where the top-most package/subsystem represents the boundary of the physical system” [74].
Model-based Systems Engineering	“Model-based systems engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation, beginning in the conceptual design phase and continuing throughout development and later life cycle phases” [39, 107].
(Model) Element	“An element is an atomic constituent of a model” [74] and “[a] model element is an element that is an abstraction drawn from the system being modeled“ [74].
Modeling Language	A modeling language is defined as “[...] a textual or graphical language used to implement one or more related types of models” [38]. “Modeling languages are generally intended to be both human interpretable and computer interpretable and are specified in terms of both syntax and semantics. The abstract syntax specifies the model constructs and the rules for constructing the model from its constructs. [...] The semantics of a language define the meaning of the constructs" [26].
Process	A “set of interrelated or interacting activities that transforms inputs into outputs” [37].
Product	A product is the “result of a process” [37].

Purpose	"An intention or aim; a reason for doing something or for allowing something to happen" [168].
Reference Architecture	A reference architecture is defined as "[...] a reusable architectural vision for use on systems within a product line or application domain" [38].
Requirement	A requirement can be defined as "[...} a statement concerning a property or the performance of a product, a process or the people involved in the process" [165]. In addition, a requirement describes a condition or capability to be provided by a system and needed by a stakeholder to solve a problem [41].
Resource	Resources describe an "asset that is utilized or consumed during the execution of a process" [8]. The resources include, for example, "funding, personnel, facilities, capital equipment, tools, and utilities such as power, water, fuel and communication infrastructures" [37].
Stakeholder	The term stakeholder describes an individual person or group of persons who have one or more concerns about the system of interest and an interest in the system meeting the needs and goals of that person or group [38, 39].
Surroundings	The surroundings describe the elements within a system environment, which are not relevant for the definition of the system and do not have any relationship to the system of interest.
System	A system consists of a fitting set of interacting elements which form a certain structure and are composed to achieve a stated purpose and goal within a defined specific environment.
System Architecture	"System Architecture is the organization of the system components, their relations to each other, and to the environment, and the principles guiding its design and evolution" [45].
(System) Context	The (system) context is "[⋯] determining the setting and circumstances of all influences upon a system" [30].
System of Interest	The system of interest describes one specific system and its life cycle as well as architecture considered during the process of preparing an description of the architecture of the system [30, 37].

Systems Engineering (SE)	"Systems Engineering is a transdisciplinary and integrative approach to enable the successful realization, use, and retirement of engineered systems, using systems principles and concepts, and scientific, technological, and management methods" [58].
Technical Solution	The technical solution describes the individual elements and the relationships between those elements, which form the architecture of a system, realize the specified requirements, and provide the necessary functions needed to achieve the system goal(s).
Tool	A "[tool] (aids) assist the representation and/or documentation of knowledge" [154].
Top-Down Approach	The top-down approach can be defined as "designing a system [...] by identifying its major components, dividing them into their lower level components, and then repeating the process until a designated level of detail is achieved" [174].
View	An architecture view is a "[…] work product expressing the architecture of a system from the perspective of specific system concerns" [30] and "[…] from a specific viewpoint and with a specific degree of granularity […]" [34] based on [30].
Viewpoint	An architecture viewpoint describes a "work product establishing the conventions for the construction, interpretation and use of architecture views to frame specific system concerns" [30].