Herausragende Masterarbeiten

Autor*in

Ricarda Schüssler

Studiengang

Software Engineering for Embedded Systems, M.Eng.

Masterarbeitstitel

Approaching Smart Interface Specifications in the Systems of Systems Context

RPTU

Distance and Independent Studies Center
DISC

Rheinland - Pfälzische Technische Universität
Kaiserslautern - Landau

Distance Study Program
Software Engineering for Embedded Systems

Master's Thesis

Approaching Smart Interface Specifications in the Systems of Systems Context

---

Provided by
Ricarda Schüssler

in cooperation with the
Fraunhofer Institute for Experimental Software Engineering IESE

First supervisor: Prof. Dr. - Ing. habil. Peter Liggesmeyer
Second supervisor: MSc. Florian Balduf

Declaration

I assure you that I have written the Master's thesis independently and have only used the sources and resources indicated and that I have identified the passages taken literally or in terms of content from the sources used.

Stuttgart, March 2023 Ricarda Schüssler

Abstract

The rapid growth of systems, both in size and complexity, combined with their distributed nature, is posing challenges for their efficient integration and functioning. Moreover, in order to achieve sustainability objectives and future goals, systems are increasingly collaborating with each other, resulting in the emergence of Systems of Systems (SoS) that are large-scale and independent. In such scenarios, multiple stakeholders and systems from different disciplines with diverse interests need to interoperate. In various domains, this trend of growing systems creates a greater need for interfaces that ensure seamless interoperability in between and within these systems and SoS.
To address these challenges, an effective method for integrating systems and SoS is required. A key to ease this integration can be the use of interface specifications to describe and specify interfaces. However, there is currently no comprehensive understanding of how to write high-quality interface specifications, nor is there a common overview of interface specification approaches.
This thesis aims to fill these gaps of documented knowledge by reviewing recent developments and best practices for interface specifications in the context of systems engineering and SoS engineering. The review was conducted through a literature review focusing on interface specifications, complemented by an analysis of existing interface specification approaches and expert interviews. The goal is to provide an overview of current interface specification characteristics and their common use cases. Based on this analysis, a usage-driven approach in the form of customised interface specification mappings was developed, which can assist in identifying an appropriate approach for specifying interfaces. In light of the increasing connectivity in our lives, the work provides a framework for better classifying and approaching interface specifications, seeking to move away from viewing interfaces as neglected elements of systems engineering, towards a more intelligent and productive classification and approach.

Keywords: Interfaces, Interface Specification, Interoperability, Systems of Systems Engineering (SoSE)

Acknowledgements

I wish to express my deepest appreciation to all who have helped me in the completion of this project. Writing a thesis is a challenging and time-consuming task, but the support I received from my family and friends was a constant source of motivation and encouragement throughout this journey.

Florian Balduf and Dr. Martin Becker from the Institute for Experimental Software Engineering (IESE) deserve special recognition for their invaluable guidance and unyielding support at every stage of this project. Without their input, discussions, ideas and suggestions, it would have been impossible to realise this thesis.

Finally, I wish to extend my appreciation to all the individuals who have contributed to this work in one way or another, either through feedback, discussions or other forms of support. Thank you all for your help and encouragement, which has made this achievement possible, particularly during the difficult times our global community is currently facing.

"When we strive to become better than we are,
everything around us becomes better too."
- Paulo Coelho

Table of Contents

Abbreviations

AAS	Asset Administration Shell
ADL	Architectural Description Language
APE	Active Phasing Experiment
API	Application Programming Interface
ARA	AUTOSAR Runtime for Adaptive Applications
CAD	Computer-Aided Design
CS	Constituent System
DaC	Documentation as Code
DIA	Development Interface Agreement
ECU	Electronic Control Unit
GfSE	Gesellschaft für Systems Engineering
GPS	Global Positioning System
HSI	Hardware Software Interface
I4.0	Industry 4.0
IBD	Internal Block Diagram
ICD	Interface Control Document
IDL	Interface Definition Language
ISO	International Organisation for Standardisation
LCIM	Levels of Conceptual Interoperability Model
MBSE	Model-based Systems Engineering
MVPE	Model Based Virtual Product Engineering
OMG	Object Management Group
PSM	Protocol State Machine
RE	Requirements Engineering
SaSIWG	System and Software Interface Working Group
SE	Systems Engineering
SoS	System of Systems
SoSE	Systems of Systems Engineering
USB	Universal Serial Bus

List of Tables

List of Figures

1. Introduction

1.1. Context

Nowadays systems grow in size and complexity. At the same time they become more distributed and are processed by several companies. Additionally, to reach sustainability goals as well as visions of the future, systems collaborate into so-called Systems of Systems (SoS) which are large-scale independent systems with their own purpose and operating together to achieve a common goal. Here, several stakeholders and systems from different domains with different interests have to interoperate. The trend of interconnected systems and devices is being embraced by all industries, including Industry 4.0, autonomous transportation and digital health. For instance, it is predicted that by 2025, there will be approximately 41.6 billion internet-connected devices, with 127 new devices connecting to the internet every second (Burton & Slowik, 2023). In addition, according to a 2021 McKinsey study, 95% of all vehicles are expected to be connected by 2030 (Hattrup-Silberberg, 2021). As these systems continue to grow, they remain responsible for human lives or large sums of money and thus require to be safe, secure and timely. Simultaneously, with the rise in complexity and connectivity, the number and complexity of interfaces also grow, making them even more crucial and ubiquitous than they are today.
Interfaces are means to structure complexity and to ease complex concepts. Interfaces can connect companies and services, can generate synergies, can improve efficiency, reuse, definition and clarity. Especially in systems engineering, interfaces need to cross different domains like software, hardware and mechanics. Although, interfaces can be a leverage to manage complexity, they can at the same time be one of the greatest dangers of a system, e.g. when thinking of potential attack surfaces. Here, each interface has the potential to result in loss of information, time, control or money. A promising method for systems and SoS architectural definition and the key to ease the coupling and integration of (complex) systems lies in their descriptions via interface specifications.

1.2. Problem Statement

When it comes to systems engineering development, the importance of interfaces is widely recognised. However, INCOSE identified a lack of interface definitions and specifications as a key problem area in systems engineering (Muscarella et al., 2020). In general, interface specifications constitute a critical tool in guaranteeing that intricate systems are precisely defined, appropriately executed and effectively maintained throughout their entire life cycle.
Currently, there is no comprehensive overview of interface specifications, nor is there a

common understanding of how to write high-quality interface specifications. A documented overview of the body of knowledge is missing, as stated by Davies (Davies, 2020). Additionally, there is no agreed-upon approach to interface specification within domains, leading most companies or working groups to develop their own methods. Given that these fundamental concepts are already absent from simpler systems engineering, it is even more important to address them in Systems of Systems engineering approaches. In this regard, different use cases may call for different interface specification approaches. However, there is a shortage of documentation on the relationship between use cases and their corresponding interface specifications.
To summarise, currently there is a lack of both an overview of interface specification use cases and a general understanding of the approaches to interface specifications in systems engineering and Systems of Systems engineering.

1.3. Research Approach

1.3.1. Research Questions

The following research questions can be derived from the problem statement and shall be discussed within the scope of this thesis:

RQ1	What are the recent developments and best practices for interface specifications in the context of systems engineering and Systems of Systems engineering?
RQ2	How are these interface specifications used?
RQ3	Which features do available interface specification approaches comprise?

These research questions will be discussed in the remaining sections of the thesis.

1.3.2. Research Method

Building upon the defined problem statement and research questions, this thesis aims to shed light on the complexities of interface specifications. To accomplish this, the thesis will develop an overview of use cases for interface specifications as well as an overview of various approaches to interface specification. The foundation of this work will be a literature review focused on interface specifications in the context of systems engineering and Systems of Systems engineering. A conceptual model will then be developed to decompose the problem at hand. The approach will begin with the examination of smaller systems engineering issues before moving on to larger Systems of Systems engineering concepts. The literature review will be complemented by analyses of existing interface

specification approaches and expert interviews. Finally, a customised approach to interface specification will be developed to conclude the research.
This thesis is a collaborative effort with the Fraunhofer Institute for Experimental Software Engineering IESE. It extends the work of the dynaSoS project, which concluded in January 2023 and focused on generating suggestions for improving system and software engineering practices, with particular emphasis on the (dynamic) Systems of Systems context.

1.4. Thesis Overview

1.4.1. Contributions

Drawing from the defined research questions and presented research approach, Figure 1 depicts the inputs and activities necessary to achieve the outputs or contributions of this thesis. The primary contributions of this work include overviews of interface specification use cases and approaches, as well as a customised interface specification approach.
In a nutshell, by reading this thesis one will not only get to know why interfaces are regarded as the stepmothers of systems engineering, but one will also learn how interfaces could be specified in a smart way to master complexity in systems as well as Systems of Systems engineering.

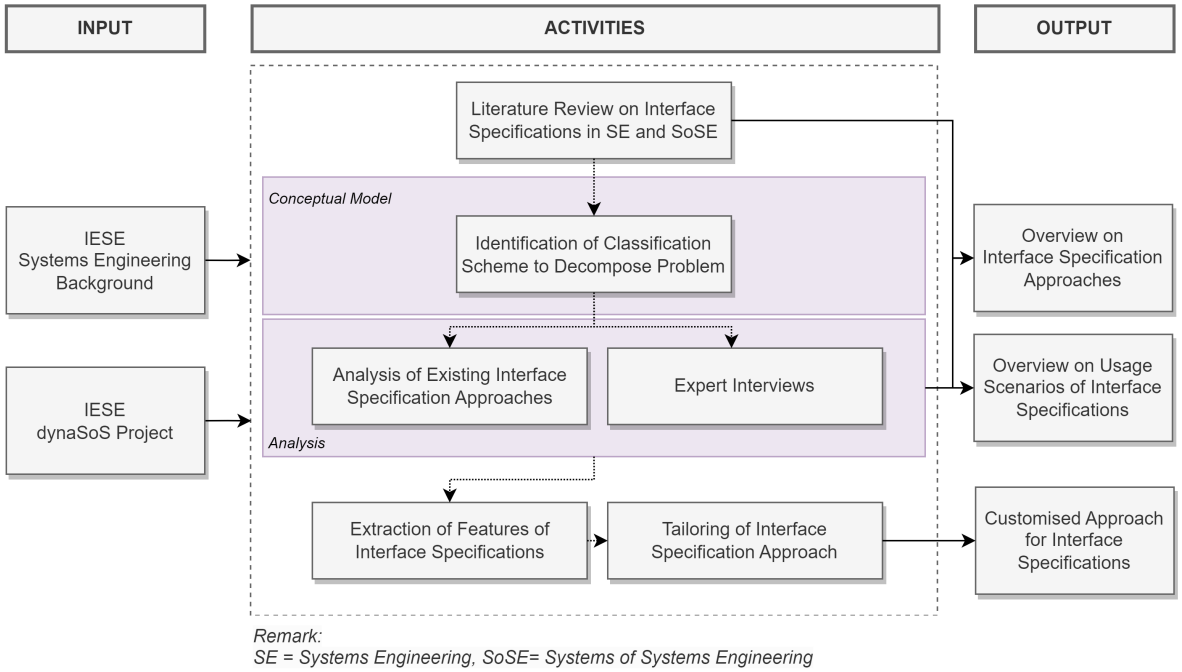


Figure 1: Thesis Big Picture

1.4.2. Thesis Structure

The remainder of this thesis is structured as follows:
Chapter 2 provides a comprehensive introduction to the fundamental concepts that are essential for the analysis of interface specifications in the context of systems engineering and Systems of Systems engineering. These concepts include interfaces, interface specifications, as well as Systems of Systems. A systematic literature review on interface specifications in both systems engineering and Systems of Systems engineering complements this, along with a description of key approaches. Building upon the foundational knowledge obtained from Chapter 2, Chapter 3 categorises existing interface specification approaches and proposes a solution approach for decomposing the general problem statement. To develop a more in-depth understanding of the current state of practice, features and use cases of specific interface specification approaches, Chapter 4 examines several examples and elaborates the results from the expert interviews. Chapter 5 presents a customised approach, along with a proof of concept. The customised interface specification approach builds on the analysis results that were achieved before. Finally, Chapter 6 summarises the results obtained and provides an outlook for future activities.

2. Foundations

This chapter presents and clarifies the fundamental concepts and terminology that are relevant and necessary for the thesis. Additionally, it aims to provide a concise overview of the literature already published on interface specifications within the systems engineering and Systems of Systems engineering context.

2.1. Key Approaches and Concepts

2.1.1. Interfaces

Interfaces serve as the foundation for interface specifications. To fully comprehend interface specifications, it is necessary to first understand what an interface is, as it is the object that requires specification. The following paragraphs cover the definition of interfaces and their various types and classifications in order to achieve this understanding.

Interface Definition The expression *interface* itself is generic in nature and is frequently used in a variety of contexts and domains. Thus, this term may both be used by someone who works in neurosurgery as well as from engineers developing spacecrafts. Etymologically it originates from the two Latin words *inter* ("between") and *facere* ("to do") which, when broadly translated, would mean "to do something between things". Within this work we will primarily focus on the term *interface* in the context of the Systems Engineering and Systems of Systems Engineering domain.
Given how general the term *interface* is, Parslov et al. (Parslov & Mortensen, 2015) have conducted a systematic literature review on interface definitions. 13 different interpretations of the manifestation of an interface were found in this review. The majority of authors believed that interfaces were either structural or functional and about half of the perceptions were thought to be part of the elements rather than a separate design object. Additionally, it was found out that different languages (i.e. systems, functional, structural languages) exist to denote the elements that interface. Parslov and Mortensen generally draw the conclusion that there is a lack of agreement about the characteristics of an interface in engineering design. They especially highlight the need for greater attention on interfaces which reside in the tension field between different engineering disciplines. This is due to the high likelihood of miscommunication caused by the lack of consensus on the nature of interfaces. Within the field of Systems Engineering (SE) and in particular within Systems of Systems Engineering (SoSE) where several engineering disciplines meet, this can have a significant impact.
To this end, INCOSE has formed a System and Software Interface Working Group (SaSIWG) in 2017 which among others shall align and integrate the terminology, processes,

methods and tools of interfaces between software engineering and systems engineering (Sheard et al., 2018). A recent survey of the SaSIWG performed by Muscarella et al. identified priority areas for improvement among the systems and software interface. The lack of interface definition, including data and specifications was ranked second as the key problem area (Muscarella et al., 2020). In order to provide a general understanding of the definitions of an interface from various sources, the following selection of definitions is presented:

- (Abbott, 1991): *"An interface is any means through which objects interact."*
- (Bachmann et al., 2002): *"An interface is a boundary across which two independent entities meet and interact or communicate with each other. All elements have interfaces."*
- ISO/IEC/IEEE 24765:2017(E) (ISO, 2017): *"1. shared boundary between two functional units, defined by various characteristics pertaining to the functions, physical signal exchanges and other characteristics 2. hardware or software component that connects two or more other components for the purpose of passing information from one to the other"*

These definitions have given a first impression on what interfaces can be regarded as. Nevertheless, despite the various definitions and questions of what an interface is and what it is intended to, one should also think about the undesirable aspects of interfaces (Davies, 2020). Davies entitles interfaces as a "someone elses problem" (SEP) since engineers only worry about the internal functionality of their system and don't look beyond its internal functions. This can lead to potential problems or failures when integrating different systems or components.

Interface Management The management of interfaces is, according to Blyler (Blyler, 2004), *"one of the most powerful tools"* of the interdisciplinary discipline of systems engineering. Software engineering as well as systems engineering nevertheless have room for improvement in the area of interface management. Several surveys and references have reported this. Namely, interface management was identified within INCOSE's SaSIWG survey as the weakest systems engineering capability based on analysis of the competencies of 50 companies. (Muscarella et al., 2020). Similarly, Wheatcraft argues that despite the known importance of interfaces one would think that a standard process exists to identify and define interface as well as interface requirements , none is in place. Each manages these activities differently due to the various cultures existing within industries and within organisations (Wheatcraft, 2010).
In contrast to INCOSE's Systems Engineering Handbook (Walden et al., 2015), which

barely mentions an interface management process, NASA's Systems Engineering Handbook describes an exemplary process (NASA, 2007). Here, the basic tasks among others are:

- Define interfaces
- Identify the characteristics of the interfaces (physical, electrical, mechanical, human, etc.)
- Ensure interface compatibility at all defined interfaces by using a process documented and approved by the project
- Strictly control all of the interface processes during design, construction, operation, etc.
- Etc.

This excerpt provides a general idea of the activities that can be included in effective interface management. However, as interface management is not the main focus of this thesis, readers are encouraged to conduct further research if they want to learn more about it. Defining and categorising the interfaces of the system of interest is a crucial first step in effective interface management, as discussed in the following paragraph that covers the topic of interface types.

Interface Types Rahmani and Thompson (Rahmani & Thomson, 2009) have highlighted that defining interfaces systematically is the first step towards standardising them. Similarly, Lynch and Sage (Sage & Lynch, 1998) argue that developing an interface taxonomy for all common interface types is necessary for a comprehensive description of an interface. In the following section, an overview of the classification of interface types based on various resources will be presented. It should be noted, however, that there is currently no universally accepted classification of interfaces in the field of Systems Engineering. Readers are advised to conduct further research on this topic if necessary.
The categorisation of interfaces into the systems that interface (Abbott, 1991) is a common method of classifying interfaces. Thus, one could distinguish between hardware/software, software/software and hardware/hardware interfaces. Since this classification revealed that software/software interface are special cases of hardware/hardware interfaces, Abbott suggests a new partitioning into structural or interpretive interfaces, symbolic interfaces and physical interfaces.
Within the NASA Training Manual for Elements of Interface Definition and Control (Lalli et al., 1997), the authors propose a categorisation into the following categories: electrical/functional, mechanical/physical, software and supplied services.
Another classification was provided by Pimmler and Eppinger (Pimmler & Eppinger, 1994) who concentrate on types of interaction in form of flows. They propose four generic interaction types: spatial, energy, information and material.

Sage and Lynch (Sage & Lynch, 1998) describe ten different interface types: internal, external, functional, physical, logical, environmental, dynamic, hardware-to-hardware, software and hardware-to-software. These different types don't seem to be mutually exclusive, so that an interface could be both external and dynamic for instance. INCOSE has pursued a similar idea with their interface classification into the categories external, internal, physical, logical interface, hardware and software interface (Benzi Lavi, 2010). The authors similarly note that one interface can be classified into more than one of the listed categories.

The International Organisation for Standardisation (ISO) as well as Wheatcraft distinguish between internal and external interfaces (ISO, 2011b; Wheatcraft, 2010). Here, external interfaces are where the system is required to interact with external systems, whereas within internal interfaces the system elements within the system interact with each other. Within OMG's modeling language SysML, interfaces are distinguished between provided and required. Here, a required interface on a port specifies one or more operations required by the block's (or its parts') behaviours. An interface provided on a port specifies one or more operations that a block (or one or more of its parts) must provide (Friedenthal et al., 2015).

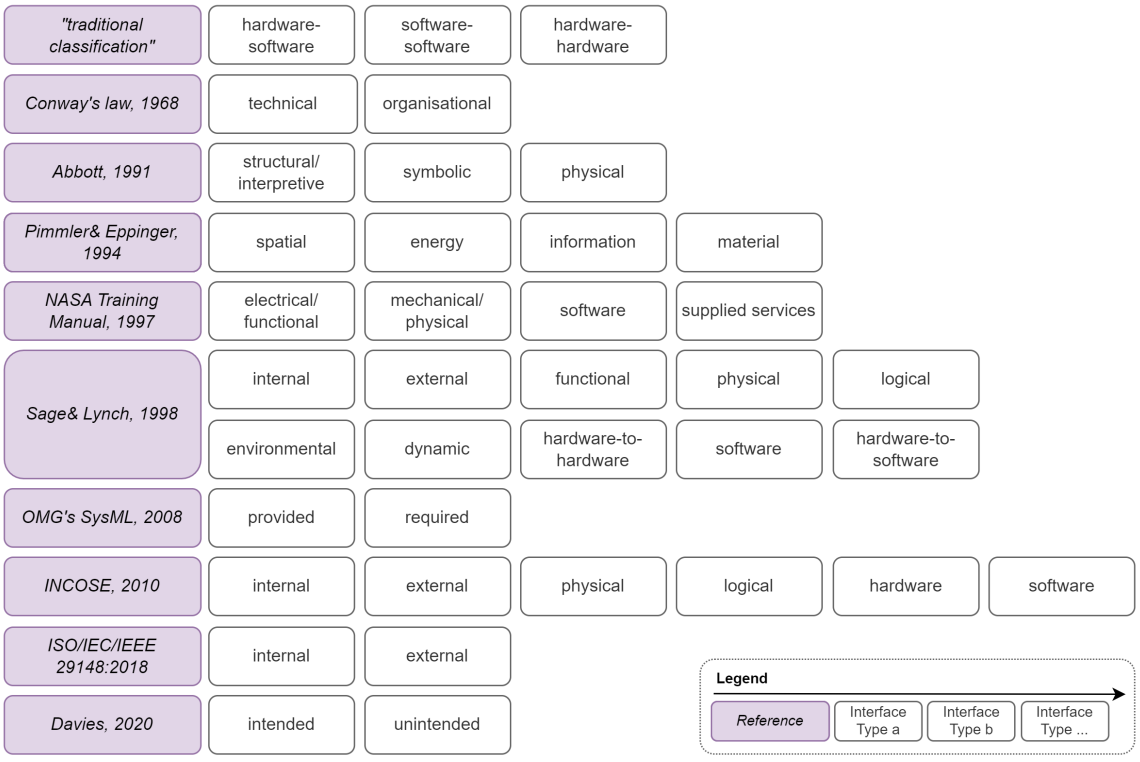


Figure 2: Interface Types Row-wise Sorted by Reference

In the context of interface classification, it is also worth to mention Conway's Law, which predicts that the communication structures within an organisation will be reflected in the

design of the systems that it produces. In other words, organisations that design systems are limited in their ability to create designs that do not mirror their own communication structures (Conway, 1968). This shows that interfaces which are based within a technical system yield interfaces on the organisational level. One example of this are interfaces that can be seen as contracts between companies.
Proceeding from the above, Figure 2 shows an overview of the possible classifications of interfaces. The different vertical classes in this case are not mutually exclusive, allowing an interface to contain one of each vertical class. The interface classifications are sorted chronologically. Interface classification is a useful approach to analyse complex interfaces by breaking them down into individual interface components. For example, a USB interface can be deconstructed into its mechanical components (such as the socket connector), electrical components and software components (including protocols and timing constraints). This approach aligns with the software engineering principle of "divide and conquer" and supports the development and maintenance of complex interfaces in Systems Engineering and Systems of Systems Engineering contexts.

Interfaces and Interoperability The term *interoperability* is frequently used in relation to interfaces, particularly in the context of Systems of Systems Engineering.
According to the ISO Systems Engineering vocabulary, interoperability refers to the *"degree to which two or more systems, products or components can exchange information and use the information that has been exchanged"* (ISO, 2017). Achieving interoperability between components requires a detailed interface specification and a comprehensive description of the components' interfaces, as noted by Louadah et al. (Louadah et al., 2014). Hause (Hause, 2018) adds that interoperability is *"the ability of making systems and organisations work together (inter-operate) for information technology or systems engineering services to allow for information exchange. Interoperability is a property of a system, whose interfaces are completely understood [...]"*. He argues that it is possible to define and specify the systems and their interfaces so that they adhere to a single communication standard. This does not, however, guarantee interoperability. Since standards are sometimes open to interpretation, two systems could both abide by the same communication standard while still being unable to communicate.

2.1.2. Introduction to Interface Specifications

As stated in the preceding paragraph, thorough interface specifications are essential for inter-operable systems. According to Blyler, interface specifications are *"critical design documents"*(Blyler, 2004). Therefore, before moving on to a more in-depth examination of these in Chapter 3, this section gives a general idea of interface specifications.

To cite the official Software and Systems Engineering Vocabulary (ISO, 2017), *interface specifications* are defined as

1. *"Description of essential functional, performance and design requirements and constraints at a common boundary between two or more system elements"*
2. *"Document that specifies the interface characteristics of an existing or planned system or component"*

The aforementioned definition portrays interface specifications as a form of description, aimed at conveying information selected by an architect to enable interaction or communication with a system and other entities. Balancing the amount of information to be disclosed about the interface within its specification poses a significant challenge, as providing too much or too little information can lead to problems (Bachmann et al., 2002). In the second place, the definition shows that interface specification are regarded as a document. For instance, in the field of avionics, this primarily textual document is frequently referred to as an Interface Control Document (ICD) (Blyler, 2004; Fosse & Delp, 2013; Louadah et al., 2014; Rahmani & Thomson, 2011). This procedure can also be found inside INCOSE's Systems Engineering Handbook mentioning that interactions between two systems are described in interface definitions, which are frequently included in a document with a title like an ICD (Walden et al., 2015). Unfortunately, the lack of a common language and terminology makes using ICDs difficult. There is no standard way of defining them, so they can vary substantially between companies (Rahmani & Thomson, 2011). Also, ICDs are currently mostly maintained manually in systems engineering (Herzig et al., 2018). Additional reasons like traceability or changeability consequently open the door to additional interface specification strategies that go beyond purely textual documents. Model-based specifications are one option (Rahmani & Thomson, 2011).
This thesis thus aims examining at how, why and when interfaces are specified and described in a "smart way", focusing not only on traditional (document-based) ICDs.

2.1.3. Systems of Systems

Definition of Systems of Systems Modern systems are expanding in size and complexity and are heading towards these so-called "Systems of Systems," which is a widespread collection of constituent systems (CS) that are large, independent systems that each serve a specific purpose while working together to accomplish a common objective (Jamshidi, 2008; Maier, 1998; Zhou et al., 2011). Thus, SoS are formed by the integration of existing systems (legacy systems), which are typically operated by different organisations, and new systems that have been designed to benefit from this integration (Ceccarelli et al., 2016). This concept is used across several application domains such as defense, automotive, energy and

health care (Cadavid et al., 2022). However, many of the traditional systems engineering principles, such as a well-defined system scope, are not applicable in a SoS environment. Table 1 presents the main differences between classical systems engineering and Systems of Systems Engineering (SoSE) as presented by Gorod and Ceccarelli (Ceccarelli et al., 2016; Gorod et al., 2008). This comparison clearly highlights the challenges that arise in

CHARACTERISTIC	SE Monolithic System	SoSE Systems of System
Scope of System	Fixed (known)	Not known
Boundaries	Static	Dynamic
Problem	Defined	Emergent
Goals	Unitary	Pluralistic
Clock Synchronisation	Internal	External (e.g. GPS)
Structure	Hierarchical	Networked
Requirements and Spec.	Fixed	Changing
Evolution	Version Control	Uncoordinated
Testing	Test Phases	Continuous
Implementation Technology	Given and Fixed	Unknown
Faults (Physical, Design)	Exceptional	Normal
Control	Central	Autonomous
Emergence	Insignificant	Important
System Development	Process Model	n/a
Tools	Many	Few

Table 1: Comparison of SE (Monolithic System) to SoSE (Systems of System)

the context of interfaces and interoperability in Systems of Systems, particularly when considering aspects such as the *scope of system*, *boundaries* and *structure*. Maier (Maier, 1998) outlined five properties that define a collection of systems as a System of System. Understanding these properties is essential in differentiating SoS from typical systems. The five properties are as follows:

- Operational Independence: Constituent systems are autonomous, acting to serve their own goals
- Managerial independence: The constituent systems may be managed independently and so can change functionality or character during the life of the SoS in ways that were not foreseen when they were originally composed
- Distribution: Components may be distributed and decoupled, with a communications infrastructure supporting collaboration
- Evolutionary Development: As a consequence of the independence of constituent systems, the SoS as a whole will change over time to respond to changing goals or component characteristics

- Emergence: The SoS exhibits new behaviour that its components do not exhibit on their own

Maier's research has identified three different types of SoS along with their characteristics. Dahmann and Baldwin later expanded on this classification and distinguished four different types of SoS (Ceccarelli et al., 2016; J. S. Dahmann & Baldwin, 2008). The various types of SoS differ in their structure and therefore may require different approaches when it comes to interfaces (Maier, 1998). These types include:

Directed SoS:	A SoS characterised by a central management structure and centralised ownership of all constituent systems (CSs)
Acknowledged SoS:	SoS where the CS are independently owned, but the owners have made cooperative agreements towards a common aligned purpose
Collaborative SoS:	SoS where independent CS interact voluntarily in order to achieve a goal that is beneficial to each individual CS
Virtual SoS:	SoS that lacks a centralised purpose and alignment

dynaSoS The Fraunhofer Institute for Experimental Software Engineering (IESE) has recently investigated **dyn**amic **S**ystems-**o**f-**S**ystems within the DynaSoS project, which was funded by the German Federal Ministry for Education and Research (Adler et al., 2022). These dynamic SoS are a specific type of SoS that share typical SoS characteristics, but are distinguished by their high dynamism and complex interdependencies among technology, society and the ecological environment. DynaSoS are characterised by frequent and rapid changes in constituent systems, context and SoS goals. Whereas traditional SoS were traditionally used in the fields of defense, national security, military, aerospace (for historical reasons), transportation and energy, the following application domains are thought to provide use cases for dynamic SoS (Adler et al., 2022; Groen et al., 2022): Smart Farming, Smart Manufacturing, Smart Mobility, Smart Healthcare, Smart Energy and Smart City and Regions. The issue of interoperability between systems is a significant challenge for dynaSoS, as noted by Adler et al. (Adler et al., 2022). Thus, in recent areas of research the topic of thorough interface specifications is of high importance.

Interfaces in SoS Systems of Systems are by definition exposed to a lot of interfaces as the integration of (new) constitutent systems plays a central role within these (Faldik et al., 2017). In fact, a SoS is essentially created by assembling and integrating, in most cases, already-existing systems to accomplish a specific task (providing goods and services in accordance with stakeholders' needs)(Bilal et al., 2014).

Thus, SoS are largely defined by interface standards (Abbott, 1991) and interface specification is a promising approach for SoS architectural definition (R. Payne et al., 2012). The following is how Maier formulates this to convey the significance of interfaces:
"When the components of a system-of-systems are highly independent, operationally and managerially, the architecture of the system-of-systems is the interfaces. There is nothing else to architect [...] The greatest leverage in system architecting is at the interfaces. The greatest dangers are also at the interfaces." (Maier, 1998).
The recommended approach for designing interfaces is to minimise their number, according to design guidelines (Wheatcraft, 2010). In the field of systems architecture, Larry Constantines' law states that systems tend to be more stable when inter-module connection or cohesion is strong, while intra-module connection or coupling is low (Stevens et al., 1974). This means that when the connectivity between parts, elements or subsystems is high, it should be hidden or "encapsulated" by the system element, making it easier to manage the discrete inter-connectivity between these elements. It is suggested that only 15% of the total possible interfaces between major subsystem components should actually exist. However, according to Boardman and Sauser, this is hard to be applied to a System of Systems (Boardman & Sauser, 2006).
Following is a choice of challenges SoS face with regard to interfaces resp. connectivity resp. interoperability (Batista, 2013; Boardman & Sauser, 2006; R. J. Payne & Fitzgerald, 2010):

- High level of complexity of the interactions so that severe architectural mismatches may occur
- Independent existence of the constituent elements originally designed for different contexts
- In a global distributed large-scale SoS, the number of interactions between the SoS components can be explosive
- Partial impossibility to foresee all connections in advance at design time, as the emergent behaviour can impact on existing connections and also demand new connections
- Constituent parts were not inherently developed to cooperate with one another
- Etc.

Early interface definition in the product development process (when possible) and the use of contracts to define and specify interfaces early and to support their validation can be strategies to limit these SoS challenges (Wäschle et al., 2021). Architectural Description Languages (ADLs) are additionally considered as a means to support these SoS specificities (Batista, 2013). More information on these will be provided in the chapters that follow.

2.2. State of the Art and Practice

2.2.1. Literature Review

To gain an understanding of the current status of interface specifications, the literature was reviewed. This section presents an overview of all activities related to this review. The review procedures largely followed the guidelines presented by Kitchenham and Charters (Kitchenham & Charters, 2007). The approach used is briefly described before listing the databases and queries that were utilised. The results of the review and their prioritisation are then presented.

Approach The research focused on conducting automated searches of digital libraries, with additional examination of cross-references found in relevant literature. To ensure up-to-date information, the search was repeated multiple times during the course of this thesis with special attention given to new additions and changes. All statements presented in this thesis are based on the latest search conducted on January 31, 2023. Before screening the research papers, specific inclusion and exclusion criteria were established. The identified content was then subjected to the following inclusion standards:

- Papers matching the search string and papers describing the approaches within the scope of this thesis.

To limit the number of results to a manageable quantity, the subsequent exclusion criteria were implemented:

- Papers not focusing on interface specification approaches for software engineering or Systems of Systems engineering.
- Papers that are not written in German or English.
- Papers that are not accessible through one of the stated databases.

Databases The following databases, recognised for having a substantial number of publications in the field of (computer) science, were searched to locate relevant publications:

- ACM Digital Library (https://dl.acm.org/)
- dblp Computer Science Bibliography (https://dblp.uni-trier.de/)
- Elsevier Scopus (https://www.scopus.com/)
- Google Scholar (https://scholar.google.com/)
- IEEE Xplore (https://ieeexplore.ieee.org/)
- Science Direct (https://www.sciencedirect.com/)

The search conducted within the Elsevier Scopus database was limited to scanning only three search fields: abstract, title and keywords.

Search Strings The subsequent search queries were utilised to identify relevant documents aligned with the research questions:

- ”interface specification” AND (”systems engineering” OR ”systems of systems” OR ”sos”)
- ”interface specification approach” AND (”systems engineering” OR ”systems of systems” OR ”sos”)
- ”interface definition” AND (”systems engineering” OR ”systems of systems” OR ”sos”)
- ”interface contracts” AND (”systems engineering” OR ”systems of systems” OR ”sos”)
- ”interface specification” AND (”use case” or ”usage scenario”) AND (”systems engineering” or ”systems of systems” OR ”sos)

The total number of search results according to the used search engines is displayed in Table 2.

SEARCH STRING	ACM DL	DBLP	ELSEVIER SCOPUS	GOOGLE SCHOLAR	IEEE XPLORE	SCIENCE DIRECT
”interface specification” AND (”systems engineering” OR ”systems of systems” OR ”sos”)	113	0	41	4960	22	804
”interface specification approach” AND(”systems engineering” OR ”systems of systems” OR ”sos”)	0	0	0	3	0	0
”interface definition” AND (”systems engineering” OR ”systems of systems” OR ”sos”)	98	0	43	4330	7	610
”interface contracts” AND (”systems engineering” OR ”systems of systems” OR ”sos”)	8	0	2	258	0	38
”interface specification” AND (”use case” or ”usage scenario”) AND (”systems engineering” or ”systems of systems” OR ”sos))	12	0	0	0	0	1

Table 2: Literature Search Results from 2023-01-31, 8:00 pm

Having performed database searches, the abstracts of relevant papers were scrutinised and the most pertinent papers were extracted based on their alignment with the research questions. This process yielded a total of 38 papers, which were subsequently assessed and categorised into three relevance grades: *”high relevance”*, *”medium relevance”* and *”low relevance”*. The outcomes were further subdivided into two categories: systems engineering and Systems of Systems engineering. An overview of all relevant papers is presented in Appendix A.1, whereas the upcoming section exclusively covers papers that were rated as *”high relevant”*.

2.2.2. Overview of Found Papers

The 19 papers that were deemed to be of ”high relevance” are listed in Table 3.

TITLE	AUTHOR(S)	PUBLISHED IN	CATEGORY	COMMENT	RELEVANCE
An Interface Specification for Principle Solutions Supporting the Cross-Domain Design of Mechatronic Systems	Stefan Möhringer,Jürgen Gausemeier.	Proceedings of DESIGN 2002, the 7th International Design Conference, Dubrovnik 2002	SE	Cross-domain interface specification in semi-formal modeling	●
Contract-based interface specification language for functional and non-functional properties	Richard J. Payne, John S. Fitzgerald	School of Computing Science Technical Report Series Newcastle University, 2011	SoSE	SysML extension for contract-based interfaces	●
Documentation-as-Code for Interface Control Document Management in Systems of Systems: A Technical Action Research Study	Héctor Cadavid, Vasilios Andrikopoulos, Paris Avgeriou	European Conference on Software Architecture ECSA 2022: Software Architecture pp. 19–37	SoSE	Benefits of replacing document-centered interface management with documentation-as-code	●
Documenting software architecture: Documenting interfaces	Felix Bachmann, Len Bass, Paul Clements, David Garlan, James Ivers	Carnegie Mellon University Pittsburg, Software Engineering Institute, 2002	SE	Approaches to interface documentation	●
Evaluation of architectural frameworks supporting contract-based specification	Richard Payne, John Fitzgerald	School of Computing Science Technical Report Series, 2010	SoSE	Surveys approaches to the contract-based specification of SoS-constituent systems	●
Interface specification for system-of-systems architectures	Richard Payne, Jeremy Bryans, John Fitzgerald, Steve Riddle	7th International Conference on System of Systems Engineering (SoSE), 2012	SoSE	Research challenges arising from combination of SysML with formal notations	●
Model-based Interface Specification for Systems Integration in Systems of Systems Engineering	Daniele Gianni, Andrea D'Ambrogio, Pierluigi DeSimone, Marco Lisi, Michele Luglio	INCOSE International Symposium Volume 22, 2012, pp. 2040-2052	SoSE	Interface Communication Modeling Language (ICML), industry: Global Navigation Satellite Systems	●
Modeling systems-of-systems interfaces with SysML	Peter Shames, Marc A. Sarrel, Sanford Friedenthal	14th International Conference on Space Operations, 2016	SoSE	SysML modeling of SoS and their interface	●
Modelling system of systems interface contract behaviour	Oldrich Faldik, Richard Payne, John Fitzgerald, Barbora Buhnova	Formal Engineering approaches to Software Components and Architectures, EPTCS 245, pp. 1–15, 2017	SoSE	COMPASS Modeling Language (CML)	●
NASA Reference Publication: Training Manual for Elements of Interface Definition and Control	Vincent R Lalli, Robert E. Kastner, Henry N. Hartt	NASA Reference Publication 1370, 1997	SE	NASA training manual for interface definitions	●
New interface management tools and strategies for complex products	Keyvan Rahmani, Vincent James Thomson	The 6th International Product Lifecycle Management Conference (PLM09). 2009	SE	Interface management model for multidisciplinary products	●
Semi-formal and formal interface specification for system of systems architecture	Jeremy Bryans, Richard Payne, Jon Holt, Simon Perry	IEEE International Systems Conference (SysCon), 2013	SoSE	Interface design pattern for interface specification	●
SysML contracts for systems of systems	Jeremy Bryans, John Fitzgerald, Richard Payne, Alvaro Miyazawa, Klaus Kristensen	9th International Conference on System of Systems Engineering (SOSE), 2014	SoSE	Contract Pattern for contractual specification of interfaces	●

Continued on next page

TITLE	AUTHOR(S)	PUBLISHED IN	CATEGORY	COMMENT	RELEVANCE
Systems engineering interfaces: A model based approach	Elyse Fosse, Christopher L. Delp	IEEE Aerospace Conference, 2013	SE	Model-based engineering approach using SysML for interfaces	●
Towards an interface description template for reusing ai-enabled systems	Niloofar Shadab, Alejandro Salado	IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2020	SE	ICDs for ai-enabled systems	●
Towards automating Interface Control Documents elaboration and management	Hassna Louadah, Roger Champagne, Yvan Labiche	ACES-MB 2014 In conjunction with MODELS 2014	SE	Approach on automation of ICDs in avionics domain	●
Towards practical and sound interface specifications	Hans Jonkers	International Conference on Integrated Formal Methods IFM, pp. 116–135, 2000	SE	ISpec approach to interface specifications	●
Verifying interfaces and generating interface control documents for the alignment and phasing subsystem of the thirty meter telescope from a system model in SysML	Sebastian Herzig, Robert Karban, Mitchell Troy, Gary L. Brack, Frank G. Dekens	Modeling, Systems Engineering, and Project Management for Astronomy VIII, 2018	SE	Method for verification of interfaces and generation of ICDs from SysML	●
Will the ICDs please stand up? An attempt to reason about subsystem interfaces in avionics system integration	Roger Champagne, Hassna Louadah	Architecture-Centric Virtual Integration (ACVI), 2016	SE	Research on establishing common vocabulary for ICDs in avionics	●

● = high relevance ◐ = medium relevance ○ = low relevance

Table 3: Literature Search Results of Highly Relevant References

2.2.3. General Findings

In essence, the majority of the identified papers are closely related to research questions RQ1 and RQ3, as they address various methods of interface specifications within the context of systems engineering and Systems of Systems engineering. However, these papers only make implicit references to the features of the approached methods. Therefore, in subsequent steps, it is necessary to extract the features from the identified approaches. Moreover, a comprehensive overview of interface specification approaches within the two domains, such as a literature review, survey or mapping study, could not be located. This fact emphasises the significant relevance of this thesis in providing such an overview. Furthermore, the discovered papers do not explicitly address the use cases of interface specifications, as requested by research question RQ2. Thus, an alternative method, other than a literature review, is required to identify the use cases of interface specifications. The upcoming two sections will elaborate on the primary papers discovered, classified into two categories: systems engineering and Systems of Systems engineering.

2.2.4. Interface Specifications in Systems Engineering

Guidelines on Interface Specifications An overview of interface specification approaches in Software Engineering is presented by Bachmann. In his paper, he describes *"ways to document an important, but often overlooked, aspect of software architecture: the documentation of software interfaces"* (Bachmann et al., 2002). For this purpose, he explains several general terms in this context as well as the stakeholders involved in interface specifications. Furthermore, he presents a standard organisation that can be followed for interface specification. His statements are supplemented by four examples of interface specifications and their descriptions. These examples include the SCR-style interface, Interface Definition Language (IDL), a custom notation and an XML interface specification.
Another noteworthy document in the field of interface specification is the "NASA Training Manual for Elements of Interface Definition and Control" (Lalli et al., 1997). This training manual provides an overview of interface definition and documentation with a claim that the presented information can significantly reduce the amount of paper and data required in interface definition and control processes, as well as shorten the time required to prepare an interface control document. The manual emphasises the importance of interface definition and control in system engineering, particularly with regards to document-based Interface Control Documents. Furthermore, it highlights the importance of selecting content for ICDs on a case-by-case basis and tailoring the level of detail shown on interface specifications according to the type and degree of design dependency at the interface (Lalli et al., 1997).

Interface Control Documents In both academic literature and industry, Interface Control Documents are frequently employed as a document-based approach to interface definition. Nevertheless, the conventional manual maintenance of ICDs can potentially compromise the integrity and accuracy of the documents over time, resulting in discrepancies between the actual state of a system's interfaces and the information documented in the ICDs (Herzig et al., 2018). Consequently, researchers are exploring alternative approaches to document-based ICDs to address this issue: Herzig et al. (2018) propose a methodology for verifying interfaces and generating Interface Control Documents from a system model in SysML, utilising the Thirty Meter Telescope as a case study. Their approach involves deriving software interfaces from the behavioural specifications of a component through static analysis of the interactions described in the behavioural specifications. This information can then be used as a basis for comparison and verification, as well as for generating artifacts such as ICDs. The authors establish a mapping between elements specified in an ICD and the SysML constructs that implement them. They

observe that while SysML provides constructs for interface specification, their application is limited as they cannot be formally linked to component behaviour, necessitating manual inspection to ensure compliance with the specified interfaces (Herzig et al., 2018).
Champagne and Louadah have identified two main problem areas in Interface Control Documents: the lack of a commonly accepted language for their definition and use and the lack of tool support for their production and consumption. To address these issues in the context of avionics systems integration, they aim to develop a common vocabulary for ICDs. However, one of the main challenges they face is the absence of concrete examples of industrial caliber (Louadah et al., 2014). Additionally, they aim to create reliable and cost-efficient mechanisms for creating and managing ICDs by using a model-driven engineering approach to extract information from current ICDs. This approach allows for different levels of abstraction and perspectives to be included in the interface specification (Louadah et al., 2016).
Shadab and Salado (2020) present an approach for documenting the essential information required in an Interface Control Document to assess the compatibility of an AI-enabled component in a system for which it was not originally designed. They draw inspiration from Google's Model Card concept, which aims to standardise the documentation of machine learning models, including intended use, limitations and ethical considerations. As a result, the authors have developed two tables summarising the crucial features and considerations that should be documented in the interface description of AI-enabled components (Shadab & Salado, 2020).

Model-based Approaches Rahmani and Thomson proposed a systematic approach for standardising interface definition and reducing incompatible design decisions. Their approach employs interface categorisation and hierarchisation principles to establish a unique interface architecture topology for two interacting subsystems. Their paper presents an interface management model for multidisciplinary products, with the objective of enhancing communication among design groups regarding product data. Object-oriented technology is used to model interfaces since it is consistent with the object nature of product data and provides a formal framework for integrating interface data into Product Lifecycle Management (PLM) software. This approach is particularly useful in managing large numbers of interfaces, such as those found in aircraft or flight simulators (Rahmani & Thomson, 2009).
Möhringer and Gausemeier proposed a modeling approach for mechatronic systems that aims to facilitate collaboration and coordination among engineers from different domains. Their approach includes a "negotiation function" that allows engineers to work independently and simultaneously while ensuring that modifications to one module are performed

in coordination with the concerned department to avoid inconsistency with another module. This approach enables effective collaboration among engineers and facilitates the design and development of interdisciplinary systems. The authors suggest that their approach is particularly useful in complex systems where engineers from different domains work on different parts of the system (Möhringer, Gausemeier, et al., 2002).
Fosse and Delp presented a model-based approach that utilises SysML to specify interfaces, which can resolve the use of multiple documents in Systems Engineering. They suggest describing the system from various perspectives to model the broad range of interfaces and interactions. To this end, they propose seven viewpoints to specify interfaces, which include identifying the interfaces of a given part and specifying constraints such as policies, agreements and performance limitations. To map interface properties to SysML properties, they recommend mapping information input/output and material input/output to flow ports in SysML. The authors utilise SysML's extensibility with a Domain Specific Language to customise SysML blocks to "Interface Specifications" and SysML operations to "interface operations." By extending SysML, they create the appropriate customisation classes, enabling the creation of properties or relationships that are true for any interface specification or interface operation, which can be reused by applying the appropriate stereotype. For example, if all interface specifications have a property that identifies the time for which the specification is valid, this property could be part of the block associated with the customisation class for the "Interface Specification" stereotype. Another extension presented by the authors is the use of SysML constraint blocks as agreements among two or more systems or components. This extension allows for the realisation of timing or quality constraints that must be met for interactions to be successful in SysML (Fosse & Delp, 2013). Similarly, Sabetzadeh et al. proposed a methodology for modeling software/hardware interfaces using SysML. Here, interfaces are treated as separate design objects and this approach is suitable for a top-down design methodology, as described in their work from 2011 (Sabetzadeh et al., 2011).

ISpec According to Jonkers, an effective interface specification should act as a binding agreement between developers and should be used for component verification or code generation. To strike a balance between producing high-quality interface specifications with minimal effort, Jonkers proposes the use of ISpec as a promising approach. Developed at Philips, ISpec is a logical framework of concepts, principles and techniques designed to improve the quality of interface specifications. ISpec is not a method or a language, but it provides six levels of specification, ranging from an Interface Definition Language (IDL) level to a formal level, to support different levels of detail and formality. Jonkers suggests that this approach can effectively raise the standard of interface specifications

while minimising costs (Jonkers, 2000, 2004).

Interface Specification Approaches in Practice As mentioned earlier, there is no universally accepted approach to interface specifications in practical applications in systems engineering (Fosse & Delp, 2013; Herzig et al., 2018; Louadah et al., 2016). Consequently, different solutions are employed depending on the specific domains or companies involved. For instance, textual specifications are utilised in various safety-related scenarios, while descriptions within models are created using SysML and the Asset Administration Shell is used in the Industry 4.0 domain. Despite an extensive literature review, no systematic literature review, survey or classification of interface specifications employed in practical applications was found. Therefore, Chapter 4 provides a more detailed analysis of this topic.

2.2.5. Interface Specifications in Systems of Systems Engineering

Within the Systems of Systems engineering domain, several papers could be found covering the broad topic of interface specifications. Especially in the context of contract-based resp. formal methods, research was done starting from the year 2010.

Contract-based or Formal Methods The scholars Payne, Fitzgerald and Bryans have published multiple papers on formal interface specification approaches in the domain of Systems of Systems engineering. Their research was primarily conducted in collaboration with the EU-funded COMPASS (Comprehensive Modelling for Advanced Systems of Systems) project, which ran from 2011 to 2014. The central motivation was the use of contract-based methods in SoS engineering due two potential benefits: they allow for the analysis of SoS level properties and provide confidence to designers that constituent systems will adhere to expected properties on interfaces (R. Payne et al., 2012). The main approaches in this area will be presented in the following.
The researchers' initial approach involved the assessment of diverse architectural frameworks to facilitate contract-based methods, as noted by Payne et al. (R. J. Payne & Fitzgerald, 2010). Their primary conclusion was to prioritise the non-functional properties of systems during the evaluation process. Additionally, the researchers conducted a high-level comparison of several frameworks, including Acme, Darwin, Wright, UML2, SysML and AADL. In the following, they proposed an extension of SysML to the so-called SysML_C to allow contract-based interfaces (R. J. Payne & Fitzgerald, 2011).
Subsequently, the authors investigated research challenges that arise when combining the Systems Modeling Language (SysML) with the formal notation VDM, using a case study from the emergency response domain as an example (R. Payne et al., 2012). VDM is

a model-oriented notation that allows for descriptions of data and functionality and is utilised to provide formal definitions of the architectural interfaces. The authors used VDM to model the SysML interfaces as VDM classes. They concluded that while this approach is a useful way to address certain issues related to independence, it is inadequate in certain respects because VDM alone could not provide all the necessary components of a formal notation. For example, the current VDM tools lack support for model checking and proof and although it supports data-based specification of functionality, VDM does not include abstractions to describe event orderings at interfaces.
Bryans, Faldik and Payne developed the Interface Design Pattern to enhance the analytical techniques available to SoS engineers by specifying both the structure and behaviour of interfaces in SoS architectures using the industry-standard notation SysML (Bryans et al., 2013). This interface definition is subsequently translated into the COMPASS Modeling Language (CML), which provides both data modeling and event ordering to create a corresponding formal model of the SoS architecture and interfaces. CML is the first modeling language specifically designed for the modeling and analysis of SoS (Woodcock et al., 2012). To achieve CML, the authors combined SysML, the previously mentioned VDM and the formal notation Citrus.

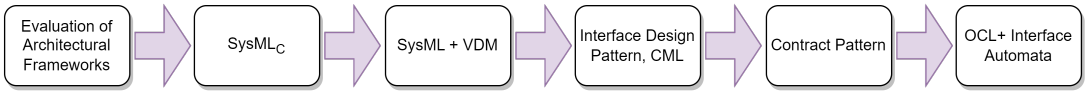


Figure 3: Evolution of Research on Contract-Based Methods in SoS

Based on this, the Contract Pattern was developed which should allow the engineer to specify constrained behaviours to which constituent systems are required to conform in order to be a part of the SoS (Bryans et al., 2014). The Contract Pattern requires five viewpoints: the Contractual SoS Definition Viewpoint, the Contract Conformance Viewpoint, the Contract Definition Viewpoint, the Contract Connections Viewpoint and the Contract Protocol Viewpoint. The approach of using CML in architectural design may have limitations in terms of adoption by domains familiar with SysML and in verifying compatibility of contracts offered by interacting CSs, which currently requires simulation or model checking.
As further deficiencies were recognised, the Contract Pattern was extended by Faldik et al. (Faldik et al., 2017) by two approaches to overcome the limitations noticed in the Contract Pattern: the use of OMG's Object Constraing Language (OCL) instead of CML and the use of Interface Automata in conjunction with SysML. An audio/video content streaming case study was used to illustrate the methodology. According to the authors, the use of OCL provides a more natural fit with the base SysML views. Additionally, the Contract Pattern was improved by separating input, output and hidden operations, while the interface

automata approach was adapted and extended by values of the contract to represent its states, allowing for verification of contract composition. To sum up, researchers have proposed and developed various approaches and possibilities in the context of contract-based methods for SoS. An overview of the approaches followed can be seen in Figure 3. Unfortunately, as of March 19, 2023, the website [1] for the COMPASS project is not accessible anymore to obtain additional information and question rises how the developed methods are used nowadays. The following sections of this thesis will explore the current state of practice and the utilisation of contract-based approaches.

Model-based methods In the context of Signal in Space of Global Navigation Satellite Systems, Gianni et al. have developed the so-called Interface Communication Modelling Language (ICML) (Gianni et al., 2012). ICML is a model-based language that can be used to graphically and unambiguously specify data interfaces, thus contributing to support the design of interoperable data communication systems. ICML is based on the standard modelling language UML. The interface specification supports verification of consistency and completeness, as well as enabling additional functions such as code generation and it includes five rectangles that address specific levels of the interface specification related to the definition of the interface's data structure for containing interface messages, including Data Definition, Binary Coding, Logical Binary Structure, Physical Binary Coding and Physical Signal (Gianni et al., 2012). Figure 4 visualises these levels.

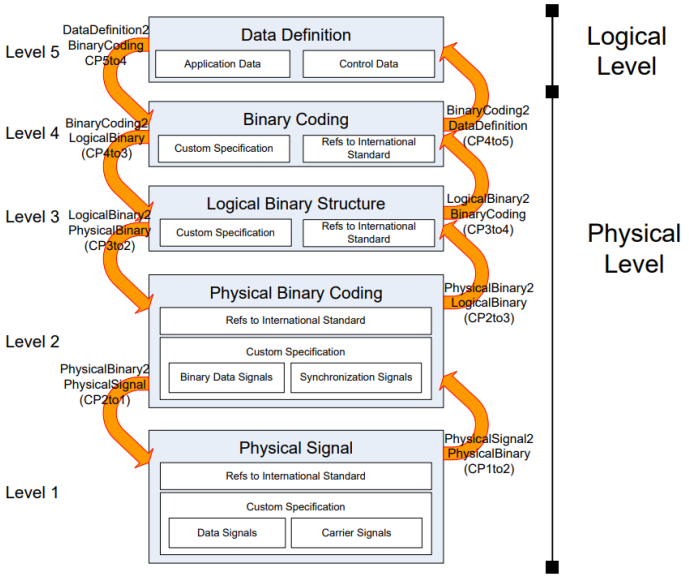


Figure 4: Structure of ICML Interface Specification from (Gianni et al., 2012)

Shames et al. (Shames et al., 2016) propose another approach to interface specification modeling, which involves developing a method for modeling space data systems and element interfaces and behaviours at increasing levels of detail in SysML. The method supports the creation of abstract views as well as detailed specification of technical interfaces and behaviour, as needed. To facilitate this, the authors introduce seven different views that

[1] http://www.compass-research.eu/

serve different purposes:

1. End-to-End view of the whole system, in context, showing data flows
2. End-to-End protocol view, showing the most significant protocols supporting the data flows
3. A “black box” view showing one element of the system and its decomposition
4. A “white box” view showing the protocol stack that implements an interface
5. The interfaces of a single protocol entity
6. A state machine showing the internal behaviour of a protocol entity
7. Specifying the standards that define interface compliance

Bryans et al. provide a contribution to the ongoing discussion about model-based interface specifications. They emphasise the current limitations of interface specification within architectural description notations, including widely used notations like UML and SysML. Despite being able to define basic signatures and specify pre- and postconditions textually, these features are not frequently utilised in practice (Bryans et al., 2013).

Documentation as Code Cadavid conducted a technical action research (TAR) study aiming to investigate on potential benefits of the documentation-as-code (DaC) philosophy as an alternative in document-centered interface management strategies (Cadavid et al., 2022). In DaC, documentation is controlled in the same way as source code in current software projects. Thus, documents need to be created and maintained as rigorously as software. It was originally created to improve well-known issues such as obsolete or unreliable technical documentation. As a result, DaC requires the usage of lightweight text-based markup languages (rather than proprietary formats and authoring tools) for the documents, in order for them to be maintained with modern (and proven) source code-oriented version control systems like Git and their accompanying collaboration and automation tools. A DaC pipeline can also ensure uniformity and increase maintainability by employing vendor-independent text-based requirements (e.g. for diagrams) that can be automatically turned into visual elements that conform to the organisation’s conventions. In conclusion, Cadavids TAR illustrated that DaC can be a promising approach to address some of the recurring issues with SoS and should be explored as an improvement of the existing document-centered ICD management approaches.

3. Conceptual Model

Proceeding from the findings of the literature review as well as from the related work in the state of practice, the following chapter shall decompose the problem by classifying interface specifications. Based on the decomposition of the main problem, a solution approach to answering the research questions defined in Chapter 1 is presented.

3.1. Classification of Approaches

The previous chapter has shown that within systems and software engineering, interfaces and interface specifications can be found in a variety of locations or circumstances. Being aware of this, it becomes clear that there might not exist *the* one solution for a good interface specification. Questions which might help to understand and classify interface specification approaches could be:

- Which stakeholder shall use the interface specification for what purpose?
- What does the originator or systems engineer want to achieve with the interface specification?
- What information shall be extracted from the interface specification?
- Should the interface specification give an overview or define all details?

For example, an interface specification aimed at defining an interface between two companies looks different than an interface specification used within a single company. Thus, it depends on several dimensions how and why an interface needs to be specified. An overview of where to find interface specification approaches shall support the classification of single approaches and shall decompose the research problem before going into detail about various interface specification approaches. The classification scheme was constructed by assessing and contrasting numerous interface specification approaches discovered through the literature review and examination of the current industry standards.
The resulting classification system, which is the outcome of this preliminary analysis, is provided below. It is based on the following six dimensions:

- Domain
- Level of Abstraction
- Interoperability Level
- Life Cycle Phase
- Use Case
- Types of Interfaces

A summary of these dimensions into which interface specifications can be classified is shown in Figure 5. Depending on these factors, interface specifications may require various

features and adopt various strategies. The idea is that an interface specification can be categorised under at least one classification of each of the six dimensions depicted in the figure. In order to simplify the reader's classification process, the individual categories of the scheme are elaborated upon in the following sections.

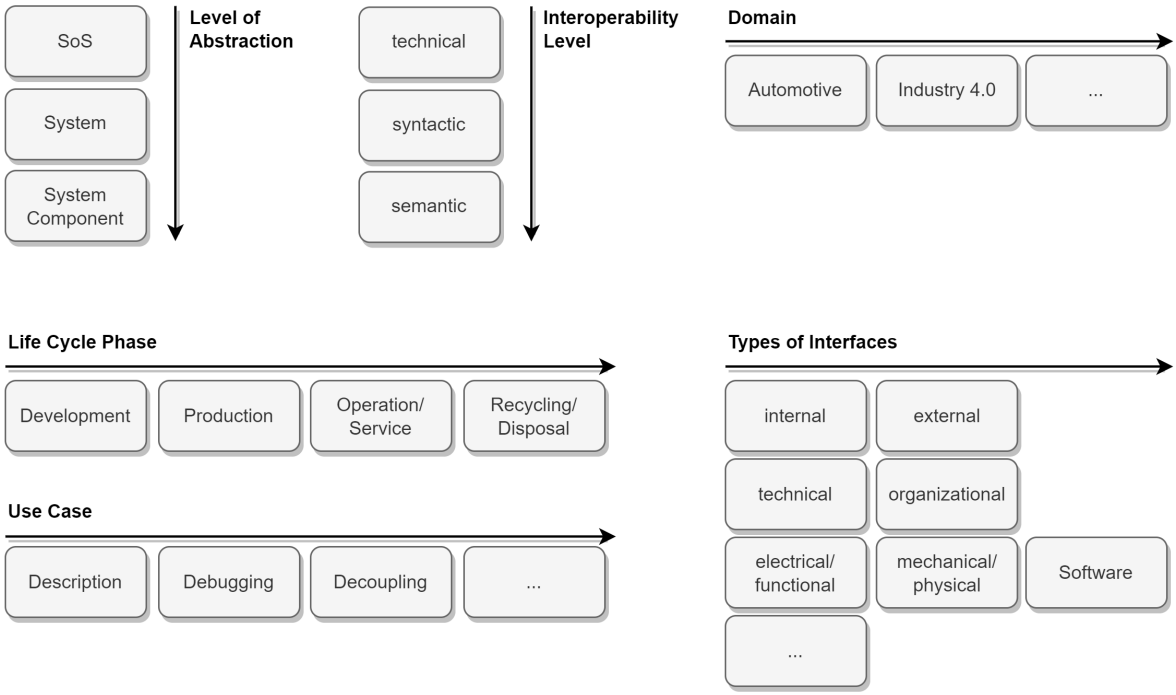


Figure 5: Dimensions of Interface Specifications

3.1.1. Domain

Different systems engineering domains adhere to various procedures, methods or standards. The following are some examples of domains: automotive, industry 4.0, aviation, healthcare, etc. Thus, it may be beneficial to understand which domain might have slightly different demands with regard to interface specifications. For instance within some domains, the usage of Interface Control Documents may still be widely accepted, while others more concentrate on model-centric approaches (Cadavid et al., 2022). As a result, it may be beneficial to consider strategies that are primarily employed in the domain of interest when integrating legacy systems. In this context, it may also be essential to bear in mind the benefits or features of different domains.

3.1.2. Level of Abstraction

The software engineering principle of "divide and conquer" is frequently employed in SE and SoSE to manage the complexity of a system by breaking it down into its constituent

parts. As a result, interfaces can be found at various levels of decomposition, ranging from the level of Systems of Systems down to the system's individual components. Depending on the level of abstraction, interface specifications may require different features. Therefore, it may be important to consider the appropriate level of abstraction when creating an interface specification. By stating that *"interfaces of all kinds should be specified at the level of abstraction appropriate for the interface"* (Abbott, 1991), Abbott substantiates this. Independent of Abbott's argument, Louadah follows the same path by bringing up various interface abstraction levels and viewpoints, including physical and electrical, data messaging and communication protocols (Louadah et al., 2014). The different abstraction

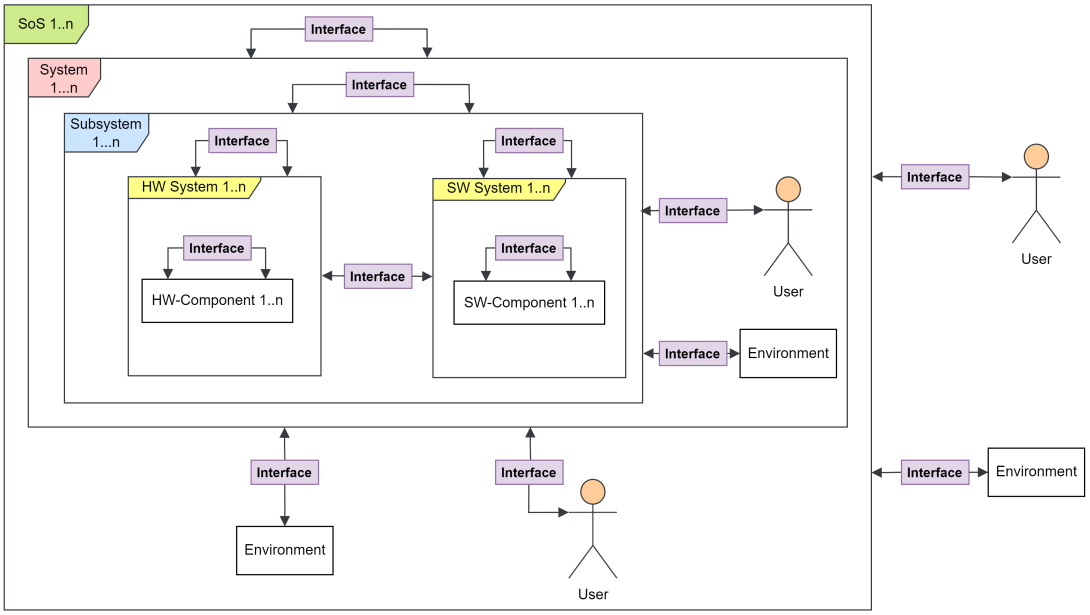


Figure 6: Exemplary Abstraction of a SoS with its Interfaces

levels are shown graphically in Figure 6, where a System of Systems represents the highest level and hardware or software components represent the lowest level. A System of Systems consists of one to n systems, with each system being composed of one to n subsystems, which are made up of hardware and software systems. Both hardware and software systems are composed of their respective components. Additionally, users and the environment can also act as actors, generating additional interfacing points.

In the figure, interfaces between all elements are depicted in a bidirectional way. It becomes clear that interfaces exist on all abstraction levels and that these interfaces presumably need to include different features or content in their interface specifications as assumed above.

3.1.3. Interoperability Level

In software and systems engineering different interoperability models or frameworks exist to explain the various levels of interoperability between systems. One prominent representative is the Levels of Conceptual Interoperability Model (LCIM) and incorporates the most technical approach (Wang et al., 2009). Other approaches for interoperability maturity models are LISI (Levels of Information System Interoperability), OIM (Organisational Interoperability Model) and EIMM (Enterprise Interoperability Maturity Model) as compared by Guédria (Guédria et al., 2008). In LCIM, six levels of interoperability are specified for identifying the interoperability needs and limitations among systems as well as for evaluating the degree of interoperability attained. It aids systems engineers in making sure that various systems can effectively collaborate to create a unified and seamless process. These six levels range from technical interoperability to syntactic, semantic, pragmatic, dynamic until conceptual interoperability. Based on this concept the INTERO model (Spalazzese et al., 2017), classifies interface specifications into three different levels of interoperability: For the simplest level "no specification", there is a communication protocol in place between involved systems. The foundational networks and protocols are defined and a communication infrastructure is established. According to LCIM, this measure corresponds to Level 1 (technical interoperability). One level above lays the syntactical dimension in which systems can communicate and exchange data without resulting in compile-time or runtime exceptions. One level above, within the semantic dimension, the information exchanged is understood by both parties and clearly interpreted to yield beneficial results.
When analysing interface specification approaches, this classification between interoperability levels shall be kept in mind. Again, depending on the approach or use cases needed, different interoperability levels might be sufficient.

3.1.4. Life Cycle Phase

Interface specifications may be necessary at different stages of a system's life cycle. It is important to consider this dimension as interface specifications may require varying features depending on the stage of the life cycle. Benveniste emphasises the challenge of providing interface specifications with sufficient richness to cover all stages of the design process in component-based embedded systems (Benveniste et al., 2018). The following approach will be based and focus on the V-Model process. Agile methods might incorporate different roles as well as phases and won't be mentioned here explicitly.
A systems life cycle can be divided into several phases, each of which requires interface specifications. Depending on the stage of the life cycle, different stakeholders are present and require different (features of) interface specifications. As a result, the use cases differ

across the life cycle phases.

Systems Engineering The traditional V-model process covers only the development cycle of a system. For reasons of sustainability, it is beneficial to not only cover the development of systems, but their complete product life cycle from initial conceptions to systems disposal. Thus, not only the design and development can be covered, but the system's production, the system's operation and service and as a last step the system's recycling or disposal. Eigner et al. developed a model for an entire product life cycle within a multidisciplinary and cybertronic environment based on VDI 2206, called MVPE model (Eigner, 2018; Eigner et al., 2017, 2019; Eigner et al., 2014). This MVPE model as well as the GREENSOFT model developed by Naumann et al. (Naumann et al., 2011) serve as the basis for Figure 7 and the further issues discussed within this thesis. Here, the MVPE model is visualised in a simplified and abstracted way with the focus on the characteristics needed within the interface specification context.

Within the development phase, the V-model process is followed. The interdisciplinary

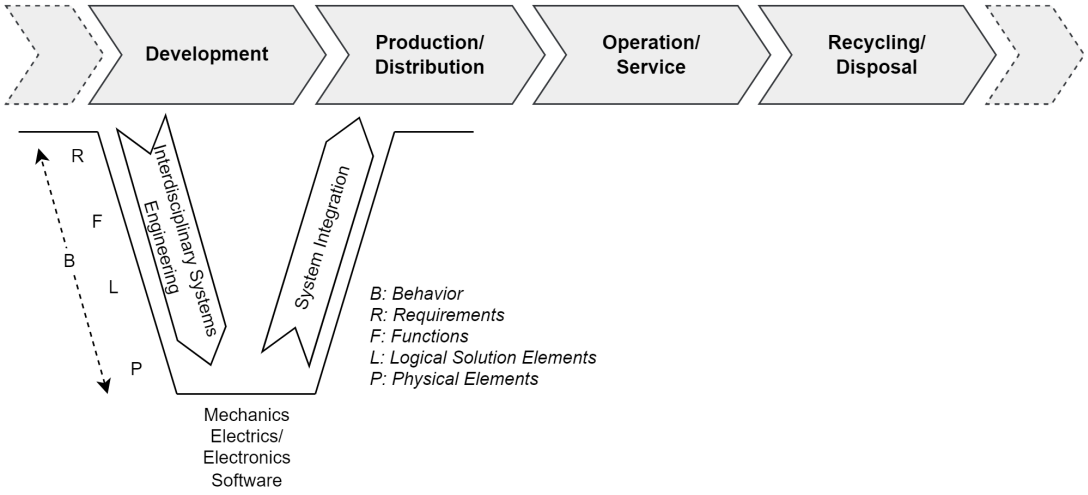


Figure 7: Simplified Systems Life Cycle Based on MVPE Model

systems engineering approach comprises the model elements requirements, functions, logical solution elements as well as physical elements. During the whole left wing of the V-model, behavioural elements are considered as well. The interdisciplinary approach is depicted by the disciplines mechanics, electrics/ electronics and software at the bottom of the V. Following the development phase, the production or distribution (in the case of software) ties in with the systems integration on right wing of the V. Followed by operation or service, the system is then used in field. Service can be understood as maintenance in this context. The recycling or disposal concludes the product's life cycle as the final stage.

Systems of Systems Engineering A "classical" system was the subject of analysis up until this point. This conventional V-model possibly cannot be used in a System of Systems a identical manner. The life cycle of cybertronic systems or SoS has been partially studied in research (Eigner, 2018; Eigner et al., 2017; Forte et al., 2021; Kozma et al., 2021). A life cycle for Systems of Systems has been defined by Kozma et al. (Kozma et al., 2021). Also Dahmann presents a generic vision of a SoS life cycle within her Double V-model. This representation, shown in Figure 8 highlights the fact that systems engineering for the individual systems in a SoS continues at the same time that SE is applied to the SoS as a whole. This can be seen as a logical interpretation of the SoSE process (D. J. S. Dahmann, 2015). As a result, various smaller systems life cycles make up a Systems of Systems life cycle. This *Double V* SoS life cycle approach will be kept in mind in the succeeding reasoning.

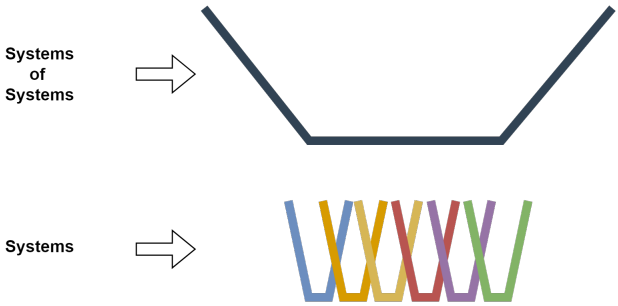


Figure 8: "Double V" Representation of SE for a System of Systems

3.1.5. Use Case

In order to comprehensively understand the specific requirements and information needs of interface specifications, it is necessary to analyse the various use cases that may require such specifications.

As with the categorisation of life cycle phases described earlier, different interface specification features are required for different miscellaneous use cases. The following focuses on identifying the use cases for which interface specifications may be required, prior to delving into the specifics of interface specification approaches in Chapter 4. The findings that are presented here were extracted from the authors as well as the supervisors experience and knowledge. It should be noted that this collection of use cases is not exhaustive and is intended to serve only as an introductory body of thought for considering the use cases for interface specifications. The use cases are presented separately for each phase of the system life cycle that was previously mentioned. Moreover, these use cases are examined separately for each stakeholder within each life cycle phase. This is because each life cycle

phase has unique stakeholders with different objectives and purposes for utilising interface specification approaches.

Use Cases during Development The first stage of the previously mentioned systems life cycle is the development of the system itself. In line with the traditional V-model approach, this development process can be further subdivided into the following process phases: requirements engineering (RE), systems design, implementation, integration and testing. Each of these phases involves different stakeholders with varying requirements for interface specifications. Moreover, a generic stakeholder known as the ”manager” is present to represent the needs of the system's company or enterprise. Depending on their particular use case, each of these stakeholders has different requirements for interface specifications. Figure 9 presents the use cases that are observable during the development phase, depending on the various stakeholders. The following provides a succinct summary of these use cases. Starting with the requirements engineer, interface specifications may

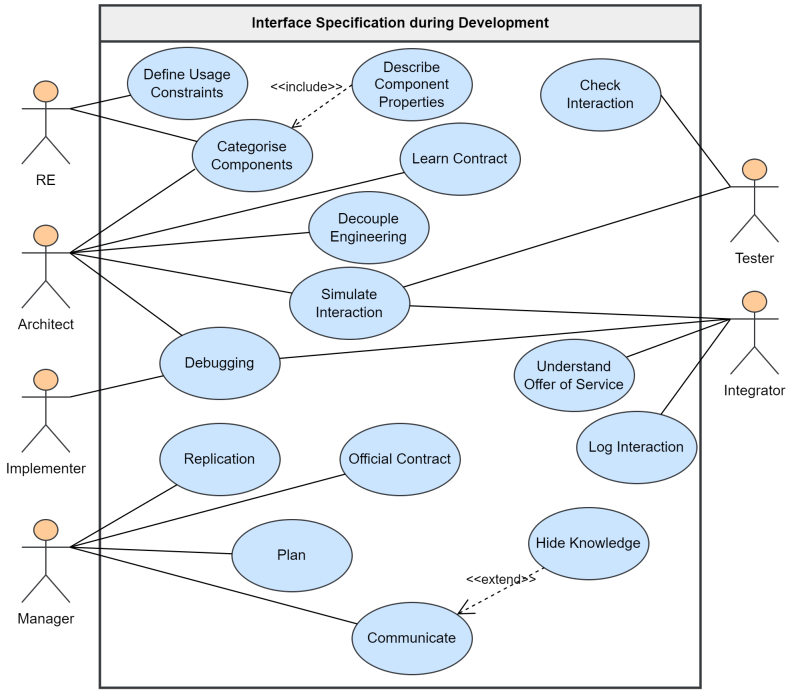


Figure 9: Uses Cases of Interface Specifications during Development

be used to define usage constraints for systems or components or to categorise these components. Additionally, interface specifications may be used to describe the properties of components. Similarly, an architect may use interface specifications to categorise components and to simulate their interactions, learn their contracts or behaviour or debug systems. Decoupling of engineering and enabling reuse of components and information hiding are made possible by well-defined interface specifications, which is encouraged in

component-based software design (Szyperski et al., 2002). Here especially the concept of low coupling and high cohesion plays a central role. Following the V-model, the implementer as the next participant in the development phase may use interface specifications for debugging purposes. The tester may need the specifications for verification and validation purposes, such as checking the interactions between components or system elements and simulating interaction where elements to test correlations are missing. The integrator may use interface specifications to debug, log interaction and comprehend component service offers.
In addition to these use cases, the stakeholder manager may require interface specifications for a number of additional purposes, including official contracting, general planning and communication and replication. Specifically, the manager may use interface specifications to conceal knowledge so that their systems appear as opaque black boxes to the outside world.

Use Cases during Production The production phase is a subsequent stage in the product or system life cycle, which builds upon the developed system. In this phase, the assembler is a relevant stakeholder who may require interface specifications for various use cases, as illustrated in Figure 10. The assembler stakeholder employs interface specifications

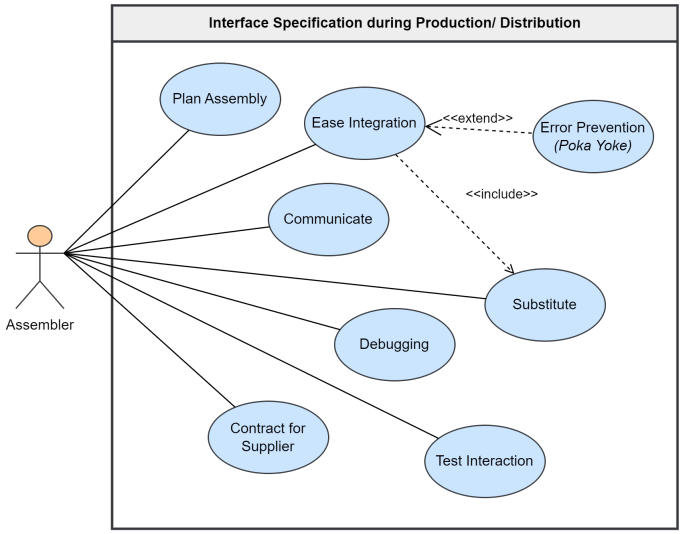


Figure 10: Uses Cases of Interface Specifications during Production

for multiple scenarios. Firstly, he may use it to plan the assembly itself or to facilitate its integration. Here, interface specifications might include the concept of ”poka yoke” which is a is a Japanese term that originally refers to mistake-proofing or error prevention in manufacturing processes through the use of simple devices or techniques. Its goal is to eliminate the possibility of errors or defects in products by preventing human errors or mistakes in the production process (Liker, 2021). Possibly this concept could be used in

the context of interface specifications.
The assembler may also use the information from the interface specifications to substitute single components, for instance, when they are unavailable. In a similar manner to the development phase, he might use the specifications to test interactions and for debugging reasons. Moreover, he may rely on it in a contractual sense and employ it for communication purposes.

Use Cases during Operation and Service In the product or system life cycle, the phase that follows production is the operation and service phase, where the terms *service* and *maintenance* are used interchangeably. This phase involves various stakeholders, including the end user, system owner, workshop and constituent system in a Systems of Systems context, as depicted in Figure 11. One of the stakeholders, the end user, may

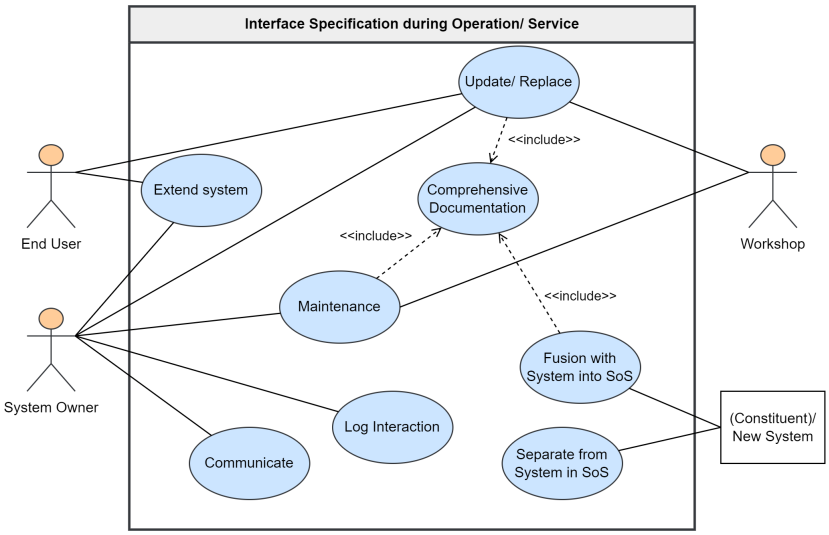


Figure 11: Uses Cases of Interface Specifications during Operation/ Service

require interface specifications to update, replace or add the necessary elements to the system. Both use cases are also appropriate for the system owner stakeholder. In addition to these use cases, the system owner may also use interface specifications for communication, logging system interactions and maintenance purposes. Interface specifications may be required simultaneously by a workshop associate for updates or replacements as well as for additional maintenance tasks. Here, interface specification approaches act as a thorough documentation for the use cases ”maintenance” and ”update/replace.” A constituent system in a context of Systems of Systems can be another stakeholder during the operational system life cycle phase. This stakeholder may require interface details for both the merging of systems into a system as a system and the separation of systems within a system of systems.

Use Cases during Disposal and Recycling The final life cycle phase of a system is reached when it enters recycling or disposal. This phase's stakeholders are similar to those from the previous phase. Nonetheless, the use cases described here are distinct. Figure 12 depicts the use cases encountered during this process step.

During this phase of the system life cycle, the end user may require interface specifications

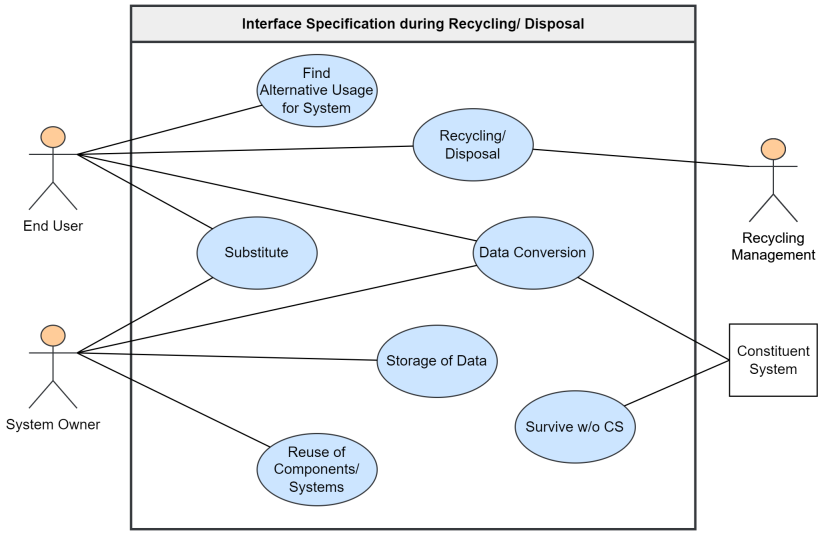


Figure 12: Uses Cases of Interface Specifications during Recycling/ Disposal

to find alternative uses for the system, to replace the system or to convert the data that is currently stored on the system. The owner of the system at the same time may aim to use interface specifications for substitution or reuse. Other options include converting and storing the system data. Interface specifications may be required by the recycling management for the recycling or disposal process itself. A Systems of Systems context's constituent systems may also require data conversion. The System of System surviving without the left constituent system is another usage scenario.

3.1.6. Types of Interfaces

Corresponding to the interface types described in Chapter 2, interface specifications might need to have different features depending on the interfaces it shall describe or specify. Here, the terms *viewpoint* or *view* play a central role. Friedenthal reasons this idea in his model-based approach as he explains that *"the viewpoint and view concept provide a mechanism to query the model for the information that addresses these concerns and present the resulting information in a way that is useful to each stakeholder"*(Friedenthal & Oster, 2017). Thus, each view addresses different contents needed by several stakeholders. To give an example, an interface specification covering interfaces between systems within a company and their internal stakeholders or viewpoints, i.e. internal interfaces, might

require different features compared to a specification that covers external interfaces, i.e. interfaces between two organisations or companies.
Chapter 2 described interface types and their classification. This classification shall be the foundation in this dimension.

3.2. Problem Solving Approach

Based on the decomposition of the problem, the following shall present a problem solving approach to the research questions defined in Chapter 1. The approach is divided into two parts which are described separately but which can content-wise be combined in the end and complement each other. Firstly, approaches to interface specification shall be analysed. Secondly, expert interviews are conducted. Details of these two parts are described in the following. Chapter 4 describes the analysis and outcomes of these activities.

3.2.1. Approaching a Customised Specification

Due to the variety of interface specification approaches, this thesis shall provide an overview before one can get into more details of a specific approach. The achievements from the previous chapter and the classification scheme introduced before, outline that *the one* interface specification solution for all needs doesn't exist. Quite the contrary holds, namely interface specifications are dependent of several dimensions and thus require different solutions. As thereby every approach has its own features, the idea is to pick to best suited features for specific use cases and to combine these into a customised approach. Bachmann substantiates these thoughts by stating that *"No single notation adequately documents interfaces; practitioners must use a combination of notations"* (Bachmann et al., 2002).
The problem solving approach that was drawn up by the author is depicted in Figure 13. In a first step, existing approaches shall be analysed. Here, it is believed that every specification approach is made of several characteristics or properties. In Figure 13, the square, triangle and circle represent these characteristics. An example for such a characteristic could be the format of an interface specification approach. The different filling at the same time shows that these very characteristics differ between the approaches *Approach 1* to *Approach n*. Pursuing the example, *Approach 1* could have a document-based format whereas *Approach 2* appears model-based. The approaches shall be fragmented into their distinctive characteristics. Out of these characteristics, their peculiar features shall then be extracted. For a document-based characteristic for instance, a feature could be the easy access or usage without special tools. The features are represented with the label F1 to F9 in the figure.
In a second step, features shall be selected based on the use cases one aims for their

interface specification. For this, a mapping of use cases to needed features or information is required.
Finally, as a third step, out of the needed features for the specific use case, a tailored interface specification approach shall be assembled. The analysis of interface specifications in the first place thus shall result in a building kit for interface specifications. This thesis

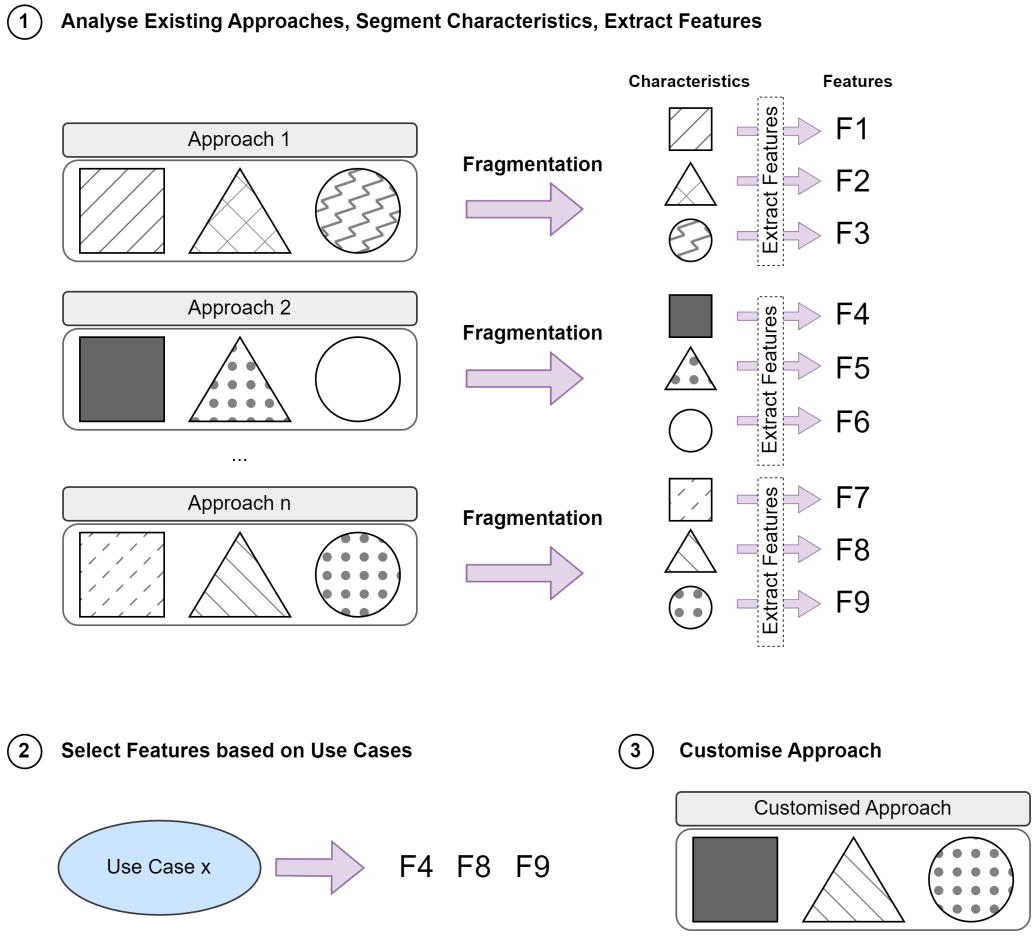


Figure 13: Problem Solving Approach

addresses both the systems engineering and Systems of Systems engineering contexts. In order to resolve the issue at hand, the approach aims to proceed from a small to large scale. Consequently, the emphasis shall be on the simpler or smaller systems engineering initially, prior to transferring concepts to the more complex Systems of Systems engineering.

3.2.2. Approach to Expert Interviews

In addition to the literature review and the analysis of approaches described before, expert interviews shall be conducted to gain additional knowledge from industry. This shall be done specifically because, as previously stated, interface specifications do not have a widely accepted solution. Experts can provide insightful information based on their

actual industrial experiences, which can be difficult to gather using other techniques, which is why this research methodology was chosen. The main goal of the interviews is to collect further data in order to identify use cases for interface specification approaches and to compile approaches themselves, including their challenges and best practices. The interviews' results shall complement the analysis of existing approaches.
The following will provide explanations of the details, broken down into the paragraphs organisation and content.

Organisation The interviews were performed as semi-structured interviews and took place over the course of about 40 to 70 minutes in an online setting using Microsoft Teams. All of the participants were interviewed in German as this was their mother tongue. Only when there was unanimous agreement among the participants, the interviews were recorded. When chosen by the participants, the company's name was concealed.
The interviewees received the questions in form of a questionnaire in advance so they could address them already beforehand. Also before interviewing, an overview of the purpose of the study was explained via a MIRO Board [2]. The MIRO board was specifically set-up for the interviews. It was used to present the current state of knowledge and provide a brief guided introduction. Given the wide range of classifications and meanings associated with "interface" or "interface specification", this board was used to direct and assist the experts while attempting not to bias them. Appendix A.2 includes the *Quick Start Guide* that was explained before every interview.
The interview was conducted in accordance with the questionnaire and data was gathered during the interview. A detailed explanation of the questionnaire's content is provided below. The responses of each interviewee to the questions were recorded and if an audio recording was available, it was later compared. The transcripts of the interviews are included in Appendix A.2.

Participants The expert interviews included five participants. To obtain a diverse range of viewpoints on the subject under discussion, the participants were chosen from diverse businesses and industries. In order to gather a variety of perspectives based on the participants' working environments, the participants were also chosen from various organisational levels and assignments. To maintain the anonymity of the interviewees' names on the one hand and to ensure clear data allocation on the other, each interviewee was given an *ID* from one to five. The experts that were interviewed can be obtained from Table 4. The experts are arranged here according to their *ID*. The table displays the specific organisations that were allowed to mention as well as differences in the positions

[2] https://miro.com/de/

and domains of the experts.

ID	ORGANISATION	DOMAIN	POSITION
ID1	Anonymous	Automotive	Group Manager Product Conformance
ID2	Robert Bosch GmbH	Automotive	Calibration Engineer Functional Safety
ID3	CLAAS E-Systems GmbH	Agriculture	Systems Engineering Coach
ID4	Anonymous	Automotive	Software System Designer
ID5	Anonymous	Industrial Plant Engineering	Technical Project Leader

Table 4: Overview of Interviewed Experts

Content Having clarified the organisational circumstances, the content-related aspects of the interviews are listed hereafter. The first step was to establish the main goal of the interview, which was to comprehend interface specification approaches and their application scenarios in the expert companies or projects. In a following step, the interview questions were particularised which resulted in four main section of the interview: introduction, interface specifications- basics, interface specifications-approach and wrap up.
In the introductory section, the characterisation of the company and projects selected for the interview were understood. The application domain, standards used in the projects as well as the software development life cycle models followed were quizzed in order to understand the general context in which interface specifications are being applied.
In the next section, the basics of interface specifications were discussed. Here, the technical part of the interview launched with questions about interface definitions as well as interface types to gain an understanding for the experts general point of view on interfaces. Furthermore, the main use cases, considered phases of life cycle and general challenges of interface specifications were inquired. Having understood the basics, the following section deals with the specifics of interface specification approaches. Detailed questions concerning the approaches appearances, specific challenges or improvement potentials were asked just like questions to interface contracts or standardisation of interfaces. Finally, the wrap up section investigates on additional improvement potentials and promising initiatives the experts apprehend in the context of interface specifications. The interview concludes with the open question which other questions the experts would have asked within the context. The research question as well as the transcripts of the expert interviews are completely listed in Appendix A.2. The next chapter will present the analysis of the interviews conducted with experts.

4. Analysis

This section presents the outcomes derived from analysing the existing methodologies employed by the industry, as well as the outcomes obtained from expert consultations, in accordance with the problem-solving approach outlined in Chapter 3. The results of both activities will be consolidated in the following chapter. Specifically, the goal is to fragment interface specification approaches into their characteristics in a first step. Subsequently, the features of these characteristics shall be extracted in Chapter 5.

4.1. Analysis of Existing Approaches

The forthcoming paragraphs will examine several examples of interface specifications. As there is currently no universally accepted industry method for specifying interfaces, this chapter will investigate some of the previously employed approaches and their distinct features. Due to the limited scope of this thesis, only a general outline of these approaches will be provided. In-depth analyses of individual approaches will need to be carried out in a separate format if necessary.
These examples were selected because they demonstrate interesting characteristics of interface specification approaches currently utilised in systems engineering and fall into various categories under the classification scheme established in Chapter 3. Each interface specification approach will be evaluated using the same framework, which is subdivided into five categories. Firstly, a classification table based on the scheme developed in Chapter 3 will provide an overview of where this interface specification approach fits within the established schema. Subsequently, the chosen approaches will be described and examined in appropriate detail under the headings of *Approach*, *Use Case*, *Features* and *Pitfalls*.

4.1.1. Commercial System: Universal Serial Bus (USB)

The analysis of interface specification approaches shall commence with the examination of commercial system specifications. In our daily lives, we rely on various systems and their interfaces, such as Universal Serial Bus (USB), AUX, Bluetooth, Global Positioning System (GPS) or metric threads, to facilitate system connections. In contrast, the worldwide wall socket represents an interface that lacks standardisation, necessitating the use of an adapter in several countries when travelling internationally. As the standardised interfaces function reliably, it is worthwhile to analyse one of them. In this context, the specification of the industry-standard USB was chosen since this interface exemplifies an interdisciplinary approach that is of relevance in the SoS context. Figure 14 shows its classification.

Classification		
	Domain	Information Technology
	Level of Abstraction	SoS Level | System Level | Component Level
	Interoperability Level	technical | syntactic | semantic
	Life Cycle Phase	Development | Production | Operation | Disposal
	Use Cases	diverse
	Types of Interfaces	diverse: mechanical, electrical, SW,...

Figure 14: Classification of USB Interface Specification

Approach The USB specification can be accessed online [3] and is available in a document-based format, specifically a PDF. The USB2.0 specification comprises 650 pages and is divided into two parts. The first five chapters provide an overview, while chapters 6 to 11 offer detailed technical information defining the USB. The detailed technical information is organised by discipline, with chapter 6 covering the mechanical part, chapter 7 dealing with the electrical part and chapter 8 covering the protocol layer, among others. The specification is written in natural English language and uses pictures and formulas where necessary for further illustration.

Use Case The USB specification does not describe any specific use cases within itself. However, it is undoubtedly utilised for several distinct use cases. Given its availability online, it can be inferred that the primary use cases pertain to communication. These could potentially involve the description of the system and its properties.

Features As the USB specification is composed in natural language and is accessible online, it can be read and comprehended by anyone interested without the need for specific tooling. The introduction highlights the relevant chapters to read based on the reader's background. Consequently, the specification's structure and format cater to a broad readership. Moreover, the use of natural language in the specification communicates semantic information. Some sections of the specification also describe behaviour, such as through state machines.

Pitfalls As the interface specification is document-based, it presents certain challenges. One such challenge is the absence of a direct link between the corresponding system data models, which results in the inability to track changes automatically. This makes change management of the specification less efficient and necessitates manual reflection of any changes. Additionally, the use of natural language in the specification impedes testing and verification processes. Furthermore, checking the completeness and consistency of the specification becomes cumbersome and must be performed manually.

[3] https://www.usb.org/

4.1.2. Development Interface Agreement (DIA)

Classification	Domain	Automotive, safety related systems (ISO26262)
	Level of Abstraction	SoS Level | System Level | Component Level
	Interoperability Level	technical | syntactic | semantic
	Life Cycle Phase	Development | Production | Operation | Disposal
	Use Cases	provide a contractual basis
	Types of Interfaces	organisational, external

Figure 15: Classification of DIA Interface Specification

Approach The Development Interface Agreement (DIA) is used within safety-related systems to specify the customer-supplier interfaces comprehensively. Specifically it outlines the safety management procedures to be followed during the development and delivery processes between the two parties. The interface specification is managed within a spreadsheet and thus is document-based. The interfaces are described in natural language. The DIA covers the safety life cycle activities to be performed by customer and supplier. It manages the responsibilities assigned to each party for each activity. The DIA serves as a communication or confirmation medium of target values and thus is an agreement between different parties within a distributed development.

		R - Responsible, S - Support, A - Approval I- Informed			
Part	Chapter	ISO26262			
2		Management of functional safety (Part 2)	OEM	Supplier 1	Supplier 2
2	5.00	**Organisation Level Safety Management**			
2	5.4.2	Safety culture	R		
2	5.4.3	Competence management(Training and qualification program)	R	S	I
2	5.4.4	Quality Management during Safety Lifecycle	R	S	
2	5.4.5	Application of the safety lifecycle(Project independent tailoring of the safetylifecycle)	R		I
2	5.00	**Project Specific Safety management**			
2	5.5.1	Organization-specific rules and processes for functional safety specific to this project	R	S	

Figure 16: Example for DIA (Embitel Technologies, 2018)

Figure 16 shows an exemplary picture of a DIA. Here, the table contains a comprehensive list of some of the required work products, along with the corresponding entity responsible for their creation, marked as 'R'. The supporting entity, marked as 'S' and the informing entity, marked as 'I', will also be listed for each work product, along with the approving entity, marked as 'A' (Embitel Technologies, 2018; Frese et al., 2019). Drawing parallels to the DIA, the field of cybersecurity engineering also implements a similar agreement, referred to as the Cybersecurity Interface Agreement, as elaborated in the ISO21434 standard. In addition, Schmittner provides an explanation of various comparable methods that are utilised throughout the different stages of the system's life cycle, including development, production and operation (Schmittner et al., 2020). This demonstrates the applicability and effectiveness of this approach within the designated use cases.

Use Case A DIA serves as a framework for effective collaboration between developers and clients by establishing roles, responsibilities and common goals. It defines safety-related responsibilities in a distributed development context and has a contractual nature as it documents the terms and conditions of the interface between multiple parties involved in the project. This ensures clarity regarding expectations, timelines and responsibilities, thereby mitigating risks associated with interface development.

Features The DIA includes features to facilitate successful collaboration between developers and clients. It defines the scope and purpose of the interface, specifies the technical details and requirements, outlines the roles and responsibilities of each party, sets timelines and deadlines and establishes procedures for communication, testing and dispute resolution. Additionally, a DIA may also include provisions for intellectual property rights, confidentiality, liability and warranties. The goal of these features is to ensure that all parties involved in the project have a clear understanding of the expectations and obligations associated with interface development and to minimise the risks and potential conflicts that may arise during the development process.

Pitfalls The DIA, being a document-based interface specification, shares the common drawbacks associated with such formats as discussed earlier (e.g., in the context of the USB approach). Notably, the DIA is a highly specific specification, requiring comprehensive understanding of its technical content.

4.1.3. Hardware-Software-Interface (HSI)

Classification		
	Domain	Automotive
	Level of Abstraction	SoS Level | System Level | Component Level
	Interoperability Level	technical | syntactic | semantic
	Life Cycle Phase	Development | Production | Operation | Disposal
	Use Cases	master complexity, traceability, parallel development
	Types of Interfaces	hardware-software interfaces

Figure 17: Classification of HSI Interface Specification

Approach The Hardware-Software Interface (HSI) refers to the point of interaction between hardware and software components in embedded systems, defining the methods of communication to enable effective collaboration. As a result, the HSI document serves as the final artifact in system development and serves as the starting point for parallel hardware and software development. The definition of the HSI requires mutual knowledge and understanding of hardware and software, achieved through a collaborative workshop

Table B.1 — Example for inputs of internal signals

Descrip-tion	HW-Identi-fier	SW-Identi-fier	Chan-nel 1	Chan-nel 2	MUX No. - Chan-nel 1	MUX No. - Chan-nel 2	Data type HW Inter-face	Address Channel 1	Address Channel 2	Unit	Interface Type	Comments	Range of val-ues	Accura-cy (% of range of values)
Inputs														
Input 1	IN_1	IN_1	x		4		U16	0x8000		V	Analogue -Internal	Analogue Input 1	0 to 5	0,50 %

Figure 18: Example for HSI from ISO26262 (ISO, 2011a)

of experts in hardware, software and systems (Macher et al., 2015; Macher et al., 2018). The HSI approach is particularly noteworthy for SoS as it adopts an interdisciplinary approach. Additionally, HSI serves as a bridge between various development domains by facilitating consensus on issues pertaining to both hardware and software development. Figure 18 shows an exemplary setup for a HSI specification based on ISO26262 (ISO, 2011a). The HSI specification is structured as a spreadsheet, containing detailed descriptions of signals, such as rotation speed, current and their relationships across different levels, including hardware, software and the overall system. It also includes information on bit configurations, addresses, conversion groups, data types, value ranges and accuracies. There are approaches that integrate spreadsheet HSI specifications with MBSE tools, as detailed by Macher et al. (Macher et al., 2015). Despite this, in actual practice, spreadsheets continue to be significant in the specification of hardware-software interfaces.

Use Case One of the primary use cases of HSI is to facilitate efficient communication between hardware and software components, which is crucial in embedded systems where these components must work together to achieve specific tasks. Here, the HSI allows for parallel development of components and aids in providing traceability. HSI is also employed to support software development and debugging, providing software developers with direct access to the hardware, thus aiding in the management of system complexity.

Features The HSI is comprehensively understandable and allows a communication of the interdisciplinary hardware-software interface between the key stakeholders. It aims to facilitate understanding and maintenance of the system and thus includes mutual domain knowledge. The features of HSI include the definition of communication protocols, data formats and control signals that are required for effective interaction between hardware and software components. Additionally, HSI provides support for error detection and recovery mechanisms to ensure reliable operation of the system.

Pitfalls The HSI specification, being a document-based interface, presents typical pitfalls associated with such specifications, as previously described in the USB specification section. In addition, the HSI approach requires careful coordination between hardware and software developers to ensure correct implementation of the interface. Specialised knowledge is necessary to develop and maintain the interface, which may pose a significant challenge for software developers lacking expertise in hardware design or vice versa.
Compatibility issues between hardware and software components are a primary pitfall of HSI. These issues can arise due to differences in communication protocols or data formats used by various components, resulting in reduced system performance or even system failures. Furthermore, systems where hardware and software are tightly integrated and difficult to separate may not be suitable for the HSI approach.

4.1.4. Asset Administration Shell (AAS)

The Asset Administration Shell (AAS) is an interface approach utilised in the Industry 4.0 (I4.0) domain. It serves as a virtual digital and active representation of an asset within an I4.0 system. Such an asset could be a machine tool for example. As the AAS is regarded as the digital copy of an I4.0 component or asset, it can provide the interface for I4.0 communication (BMWi, 2020) and shall be elaborated in the following. The AAS is classified according to its attributes, as depicted in Figure 19.

Classification	Domain	Industry 4.0
	Level of Abstraction	SoS Level | System Level | Component Level
	Interoperability Level	technical | syntactic | semantic
	Life Cycle Phase	Development | Production | Operation | Disposal
	Use Cases	diverse
	Types of Interfaces	diverse: e.g. external, technical

Figure 19: Classification of AAS Interface Specification

Approach The idea that the information world encloses the asset like a shell is the foundation of the term "Asset Administration Shell". Figure 20 demonstrates this (Plattform Industrie 4.0, 2022). Together, the physical asset and its corresponding AAS comprise an I4.0 component. The AAS functions as both the interface to the asset and the central repository for all information related to the asset, providing the interface for I4.0 communication. Through the AAS, all the properties and functions of the asset can be accessed (ZVEI, 2016). The AAS is comprised of two parts, namely a header and a body. The header contains the identifying information of the AAS and the assets represented by it, while the body houses the submodels that detail the characteristics of the AAS.

Use Case The AAS has different use cases, a few of them shall be listed in the following: The AAS can provide access to real-time data about the asset, including its current state and any potential issues. This can enable predictive maintenance, where the asset's health can be monitored and maintenance can be scheduled before a failure occurs, reducing downtime and maintenance costs. It can furthermore be used to remotely monitor and manage assets in different locations, enabling real-time access to asset data and allowing for remote diagnostics and maintenance. An AAS can also be used as a digital twin of the physical asset to allow for simulation and testing of different scenarios without affecting the physical asset. Furthermore, it can serve as a central touchpoint to an asset as it can include all of its relevant documentation.

Features A distinctive feature of the AAS is the defined semantics. Information is deposited with the help of attributes whose semantics are defined in a way that the system standardises the relevant information and makes it available via the respective RAMI4.0 layers. The attributes are e.g. standardised via the ECLASS standard. ECLASS is a global reference data standard for the classification and unambiguous description of I4.0 products and services.

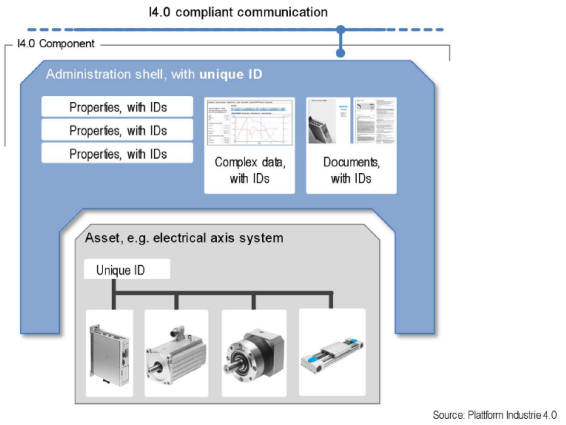


Figure 20: Example for AAS (Plattform Industrie 4.0, 2022)

It can connect components from different manufacturers using this feature because they all have the same definition of their properties and operations.(Standardization Council I4.0, 2022). By utilising this standardisation, components from different manufacturers can be connected with ease, as they share the same definition of their properties and operations. As a result, the AAS provides a standardised and harmonised interface to assets and counteracts the proliferation of various data or protocols within domains.
Currently, information on a product's functions is mostly available in (document-based) handbooks, which the AAS digitises. The AAS shall serve as an interface throughout the product-, system- and component levels. It is a machine-readable, technology- and device-agnostic description of a component. Furthermore, the concept of the AAS shall make life cycle independent and standardised information exchange possible (BMWi, 2020). There are currently three types of Asset Administration Shells distinguished (BMWi, 2020):

- Passive AAS (Type 1): serialised files, e.g. XML/JSON, contain static data that

can be distributed as files
- Re-active AAS (Type 2): contain static data which can be accessed via a standardised Application Programming Interface (API) as the interface
- Pro-active AAS (Type 3): can interact with other AAS via an I4.0 communication language (interoperable interface)

These three AAS types provide different concepts with regard to interface specifications and contribute to the interface specifications' overview that follows.
Further investigation is currently being carried out in the context of AAS and interfaces using the Thing Description (TD) framework of the Web of Things (WoT) as a submodel to develop an interface description. The objective is to depict the interface of any entity through a standardised TD that encompasses consistent communication technology, nomenclature and unambiguous semantics (Pakala et al., 2021).

Pitfalls The AAS concept presents intriguing approaches towards interface specifications in general. This contributes to the broader comprehension of interface description and management possibilities. However, the complete AAS concept necessitates substantial standardisation efforts and therefore its applicability to other domains or smaller use cases is not straightforward. Additionally, communication between assets in the AAS context utilises various protocols. Moreover, several critical papers exist that scrutinise the AAS concept, such as Reich et al., who advocate for keeping simple concepts simple in the digitalised interconnected world. Additionally, at present, the definition and content of submodels is not entirely clear (Reich et al., 2021) and submodels for certain use-cases are currently being created.

4.1.5. AUTOSAR adaptive

AUTOSAR's goal is to create an open industry standard for automotive software architecture among suppliers and manufacturers (Heinecke et al., 2006). Thus, AUTOSAR among others focuses on interfaces and serves as a further input to current interface specification approaches in this thesis. In addition to the classic AUTOSAR standard, AUTOSAR adaptive focuses on use-cases such as highly automated driving or wireless interfaces to components outside the vehicle, like vehicle-to-x-applications. As these use cases foster even more and possibly more complex interfaces, the differences between AUTOSAR classic and AUTOSAR adaptive as well as the add-ons that are needed within the adaptive context, shall be elaborated in the following. The classification of AUTOSAR adaptive interface specifications is shown in Figure 21.

Classification		
	Domain	Automotive
	Level of Abstraction	SoS Level | System Level | Component Level
	Interoperability Level	technical | syntactic | semantic
	Life Cycle Phase	Development | Production | Operation | Disposal
	Use Cases	e.g. communication beyond vehicles themselves
	Types of Interfaces	external, SW

Figure 21: Classification of AUTOSAR adaptive Interface Specification

Approach Generally, as applications are an electronic control unit's (ECU) competitive advantage, AUTOSAR doesn't standardise an application's functional internal behaviour (e.g. algorithms, optimisation), but rather the content exchanged between applications (Fürst et al., 2009). This content can be exchanged between several OEMs and suppliers. For all five vehicle domains- Body and Comfort, Powertrain, Chassis, Occupant and Pedestrian Safety and HMI, Telematics and Multimedia- the AUTOSAR release 4.0 includes a collection of application interfaces that AUTOSAR has standardised. With software reuse and exchange being one of the primary requirements of AUTOSAR, efforts have been concentrated on interface specification of well-established applications. One of the most important aspects of application reuse is the use of AUTOSAR Standardised Application Interfaces. The Application Interface Table includes information that has been standardised by professionals from all partners and members. Here, interfaces were standardised in terms of both, syntax and semantics. Software developers and implementers can use these standardised interfaces to expand or reuse software components without being restricted to a particular hardware or ECU. The Application Interface Table has the format of an Excel table and contains a number of work sheets (AUTOSAR, 2022b; Fürst et al., 2009). These sheets handle the following key pieces of information that are pertinent to the application interface: compositions (main compositions result from the five domains and the connection to the human machine interface), components, ports, port interfaces and its descriptions of information items (VariableDataPrototypes), data types for VariableDataPrototypes, units, instances of component types and keywords.
The AUTOSAR classic platform primarily realises control units that directly access sensors or actuators and which need to fulfill hard real-time constraints. In addition to that, the AUTOSAR adaptive platform aims on applications with distinct performance requirements like highly automated driving. It extends the AUTOSAR classic platform due to the need for communication beyond vehicles themselves. Thus, the adaptive platform offers a flexible integration environment which distinguishes itself through its update ability and expandability. It can dynamically link services and clients during runtime (AUTOSAR, 2022a; Reichart & Asmus, 2021; Tischer, 2018). To achieve this, different approaches with regard to interfaces need to be taken among others. Both, the AUTOSAR classic

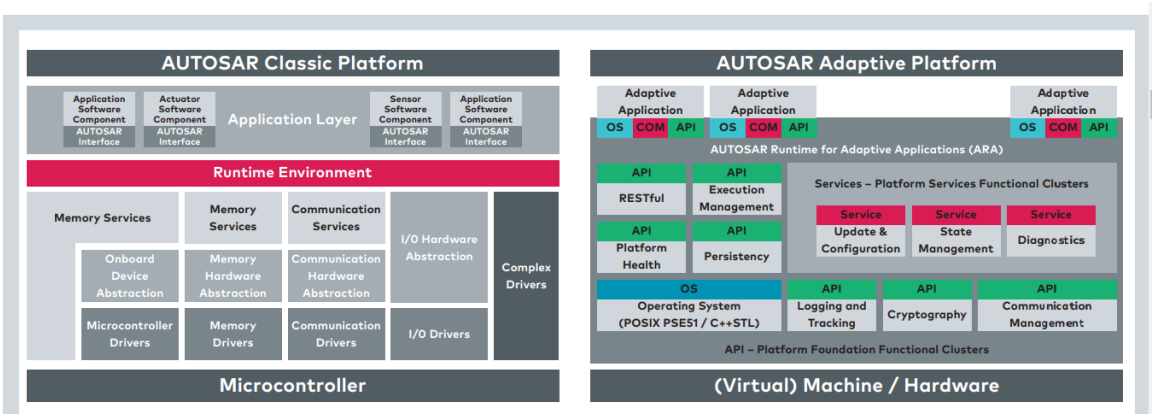


Figure 22: Overview of AUTOSAR classic and AUTOSAR adaptive platform (Embitel Technologies, 2020)

platform, as well as the AUTOSAR adaptive platform are depicted in Figure 22. The comparison of both platforms shows that for the AUTOSAR adaptive platform, a new AUTOSAR Runtime for Adaptive Applications (ARA) was introduced. This provides the programming interfaces and a communication middleware required to integrate different applications in the ECU via Application Programming Interfaces (APIs) (Reichart & Asmus, 2021).

Use Case AUTOSAR interface specifications have a primary objective of defining the communication between software components in distributed systems. They facilitate a standardised interface between different software components and enable the integration of software components from different vendors. These specifications also allow for flexibility and adaptability within the system, permitting changes to the software components without disrupting the rest of the system. Furthermore, they enable communication between software components running on different hardware platforms, thus promoting interoperability. Here, especially AUTOSAR adaptive is used to be adapted to the changing requirements of a system as it defines a set of flexible and dynamic interfaces. These interfaces are designed to support a wide range of data rates, bandwidths and latencies, which are required for highly complex and dynamic systems.

Features Generally, industry standards like AUTOSAR or Integrated Modular Avionics (IMA) in the avionics domain have paved the way into component-oriented systems, with clear interfaces and dependencies between the components. Specifically with respect to AUTOSAR, Table 5 highlights the differences between the AUTOSAR classic and AUTOSAR adaptive platforms, focusing on interfaces and their specifications. Through these differences, the adaptive platform allows adaptive applications to be integrated or replaced at runtime. While in the classic platform the complete control unit code needs to

be replaced, the adaptive platform provides the possibility to remove, update or add single applications. Furthermore, different software can be developed and distributed for an ECU, independent of each other. The applications make use of the brokering service provided by the platform without worrying about the underlying communication protocol. The ARA as the runtime environment provides standardised interfaces to efficiently integrate different applications into the system. Data exchange can happen between local and remote as well as with the AUTOSAR adaptive services (AUTOSAR, 2022a; Reichart & Asmus, 2021; Tischer, 2018).

CHARACTERISTIC	AUTOSAR CLASSIC	AUTOSAR ADAPTIVE
Communication	Signal-based („hard-wired“) - CAN	Event-based, service-oriented - Ethernet
Dynamic Updating	Not available	Incremental deployment and runtime configuration changes
Level of Standardisation	High - detailed specifications	Low - APIs and semantics
Operating System	Bare board	POSIX (provides a standard interface for communication between the application and the operating system)
Programming Language	C	C++
Use Cases	Embedded systems	High performance computing, communication with external resources and flexible deployment

Table 5: High-level Comparison of AUTOSAR classic and adaptive w.r.t. Interfaces

Pitfalls The AUTOSAR network is based on a development partnership of different automotive interested parties like OEMs, suppliers or tool providers. Based on consolidations of the partner companies, an automotive open system architecture standard to support the needs of future in-car applications was created. Thus, to achieve these standards, a high level of consolidation and agreement between the partners is needed. This high degree of standardisation might be hard to achieve in other domains and thus might be a pitfall when thinking about the carryover of AUTOSAR's basic approaches to further domains. Nevertheless, further partnerships are formed to align on standardised or common exchange formats to ease interoperability. The Sensor Interface Specification Innovation Platform (SENSORIS) is an open group of actors from the global vehicle industry, map and data providers, sensors manufacturers and telecom operators (SENSORIS Innovation Platform, 2017). They joined forces to define an appropriate interface for exchanging information between the in-vehicle sensors and a dedicated cloud as well as between clouds. These examples might show that standardisation and consolidation is an essential part of defining and specifying interfaces. This at the same time can be a pitfall as conventions, coordination and consensus might be complex to achieve.

4.1.6. Model-Based Systems Engineering (MBSE) Approaches

Given that model-based systems engineering (MBSE) is a commonly used methodology in systems engineering, it may be interesting to investigate the potential means of specifying interfaces within such systems. To facilitate this analysis, two handbooks that employ MBSE and SysML for systems engineering will be evaluated as a foundation.
The initial source of reference is a handbook written by Friedenthal and Oster (Friedenthal & Oster, 2017). The handbook presents an MBSE methodology for spacecraft architecture utilising the No Magic Cameo Systems Modeler® tool. Refer to the left section of Figure 23 for its categorisation.

Classification	Domain	Aerospace	Telescope
	Level of Abstraction	SoS Level | System Level | Component Level	SoS Level | System Level | Component Level
	Interoperability Level	technical | syntactic | semantic	technical | syntactic | semantic
	Life Cycle Phase	Development | Production | Operation | Disposal	Development | Production | Operation | Disposal
	Use Cases	diverse: documentation, development of system	diverse
	Types of Interfaces	diverse: external/ internal; mechanical, electrical,...	internal/ external; mechanical, electrical, optical, software

Figure 23: Classification of MBSE Spacecraft/ Extremely Large Telescope Interface Specifications

The second reference utilised as a basis for analysis in the MBSE context is the "Cookbook for MBSE with SysML" developed by the *Gesellschaft für Systems Engineering (GfSE)* in conjunction with the *European Southern Observatory (ESO)*. The example employed in this reference is the "Active Phasing Experiment technology demonstrator for the future European Extremely Large Telescope", which is an interdisciplinary, high-tech optomechatronic system in use at the Paranal observatory (Karban et al., 2018). Its categorisation is depicted on the right-hand side of Figure 23.

Approach Friedenthal's handbook (Friedenthal & Oster, 2017) presents a MBSE method for specifying interfaces with an example model of a spacecraft. The model's Internal Block Diagram (IBD) contains both internal and external interfaces, represented by ports. These ports are labeled according to their type, such as mechanical or electrical interfaces (e.g., mechanical i/f, electrical i/f, etc.) and are highlighted in red in Figure 24. Here, these ports enable interface specification. The spacecraft has a diverse range of item flows and interfaces, including physical and information flows. The level of abstraction in representing these flows can vary from named ports to detailed models of communication protocol stacks. However, not every element in the model is interconnected with a port. For instance, the sun in Figure 24 does not have a port unless it is necessary to specify information about the radiation on its surface.
Karban et al. (Karban et al., 2018) expand on the previous elaborations with their

example of a thirty-meter telescope. They consider interfaces to be important architectural elements at the system level, primarily because interfaces contribute to the complexity of the telescope. They discuss three concepts available for modeling interfaces, which include ports, service interfaces and blocks. Their focus is on ports and they distinguish between two types: standard ports, which are commonly used in client/server and command and control system scenarios and flow ports, which model the continuous flow of an item, whether it is data or material. Standard ports may have a state machine protocol associated with them to define the method call protocol. The continuous flow of an item can be modeled using flow ports. A flow port serves as an interface point that determines the input and output items that can flow between a block and its environment, facilitating the interaction between them. The specification of the types of data, material or energy that can flow through a flow port is established through typing. This typing can either involve a single type (such as block, value type or signal) for an atomic flow port or a flow specification that lists multiple items for a non-atomic flow port. A variety of ports is utilised in Karbans IBD, including standard ports, atomic flow ports (indicated by a single arrowhead) and non-atomic flow ports (indicated by two arrowheads).

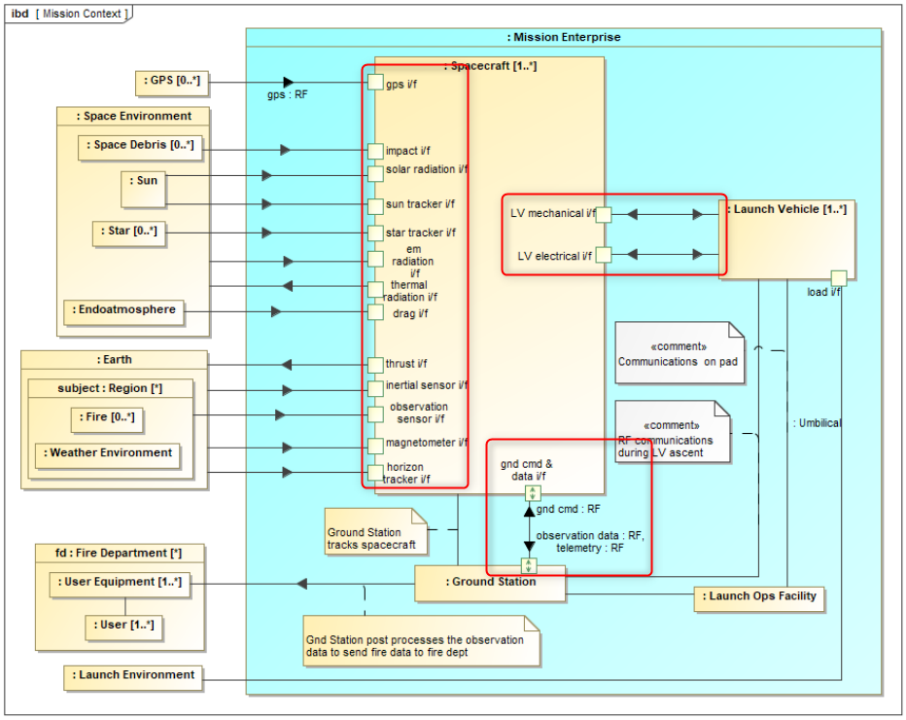


Figure 24: IBD of Spacecraft with Interfaces Represented by Ports (Marked in Red) (Friedenthal & Oster, 2017)

Use Case Although the two handbooks do not provide a specific description of the use cases of interface specifications in a model-based way, there are several use cases that exist. These could include, but are not limited to: improving systems understanding by representing interfaces in a system model, facilitating collaboration among developers,

detecting interface issues, verifying or validating interfaces, generating code automatically and integrating systems.

Features Friedenthal and Karban implicitly show several features of model-based interface specifications. The features mentioned by Friedenthal include: Model-based approaches can refer to ICDs that already specify interfaces so that they don't need to be modeled again. For this, value types, stereotyped <icd>, are used to represent the documents (Karban et al., 2009). Vice versa ICDs can be generated from the model by querying the system model and rendering the information in the designated document format in order to reach various stakeholders (Friedenthal & Oster, 2017; Herzig et al., 2018). Karban expands on these features by introducing the concept of portgroups, which can group multiple interfaces together and serve a port specification. The aim of portgroups is to group interfaces to define physical connectors and flows at part borders, which can conceal the internals of a block and enable reuse. Additionally, it is possible to define quantifiable system interfaces using SysML constraints, as demonstrated by an example from APE in Figure 25. In this example, the ≪constraint≫ *NasmythInstrumentSpecification* imposes limitations that compare the characteristics of APE to those of the telescope. One of the telescope's properties is an *allowedMass* value of 8000 kg. The constraint stipulates that the actual mass, or *realMass*, of APE should not exceed the allowed mass. In this case, the real mass of APE is set to 5000 kg.

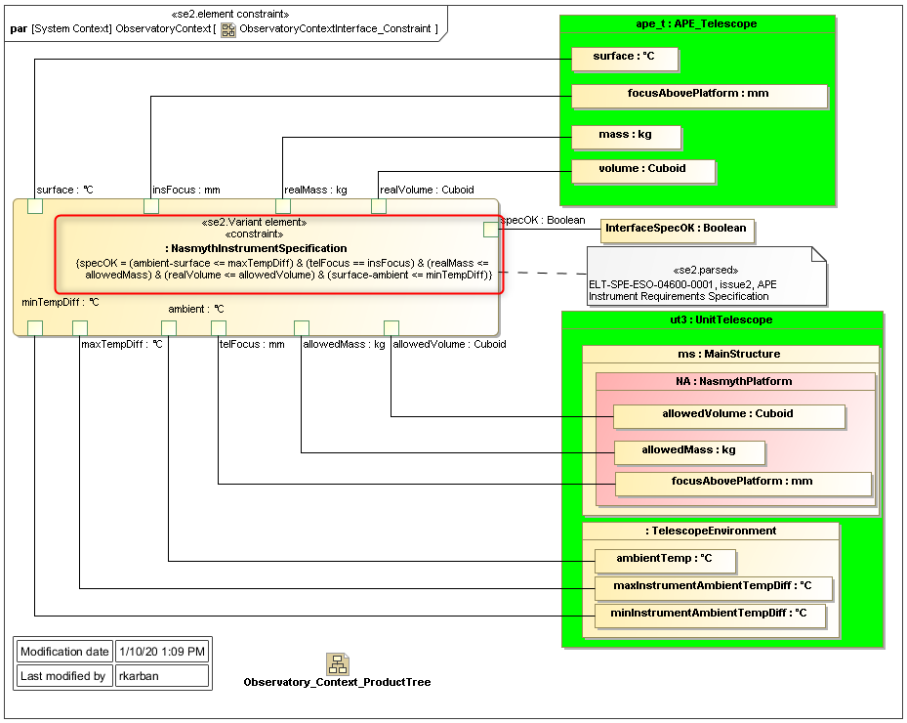


Figure 25: Interface Modeled as Constraint in APE (Karban et al., 2009)

Attributes can also serve as placeholders for rules on how to connect different interface types, such as allowing only male or female connectors of the same type to be connected,

which can be interpreted or verified by an engine.
In conclusion, the above examples demonstrate that a model-based approach can enhance the validation and reusability of requirements and system interfaces beyond mere textual descriptions. Additionally, a model can serve as an effective communication tool, helping to avoid misinterpretation issues that are often encountered when relying solely on natural language or free-format diagrams (Karban et al., 2009). Further details will be discussed in Chapter 5.

Pitfalls When it comes to interface specifications, MBSE methods based on SysML v1 have several shortcomings. Although interfaces encompass both functional and physical aspects, SysML v1 primarily focuses on functional interaction and lacks the ability to specify physical interaction points according to Weilkiens (Weilkiens, 2019). However, the upcoming version of SysML, SysML v2, is expected to address this and other issues by offering enhanced interface modeling capabilities. These capabilities include the ability to model different interface abstraction levels (such as logical, functional and physical) and interface layers (such as mechanical, electrical and software), define software protocols, physical geometries and other types of interaction controls and have complex structures. SysML v2 also allows for nested interface definitions and is set to support the generation of Interface Control Documents (ICDs) (Weilkiens, 2019).

4.1.7. Interface Definition Language: Franca IDL

Classification	Domain	diverse
	Level of Abstraction	SoS Level | System Level | Component Level
	Interoperability Level	technical | syntactic | semantic
	Life Cycle Phase	Development | Production | Operation | Disposal
	Use Cases	diverse
	Types of Interfaces	Software interfaces

Figure 26: Classification of Franca IDL Interface Specification

Approach Interface Definition Languages (IDL) are technology-neutral and language-independent in contrast to interfaces defined using a general-purpose programming language (for example, a Java interface or a C++ class). Thus, they are designed to facilitate communication between heterogeneous systems, including those written in different programming languages, running on different operating systems and using different communication protocols. Furthermore, IDLs can be classified into Beugnards classification scheme of interface contracts where he differentiates four levels of interface contracts (Beugnard et al., 2010; Beugnard et al., 1999), refer to Table 6. Here, IDLs are represented in Level

1 as they specify among others the operations a component can perform and the input and output parameters each component requires. Prominent examples of such an IDL are CORBA IDL or OMG's IDL (Object Management Group, 2018). In the following, an IDL

LEVEL	DESCRIPTION	CHARACTERISTICS
Level 1	Syntactic level	Interface definition language; specify the operations a component can perform, the input and output parameters each component requires and the possible exceptions that might be raised during operation; examples: Interface definition languages (IDLs), typed object-based or object-oriented languages
Level 2	Behavioural level	Design by Contract pattern, specify an operation's behaviour by using Boolean assertions: pre- and postconditions; examples: Eiffel, Blue, Sather, UML/OCL, iContract for Java
Level 3	Synchronisation level	Specifies the behaviour of objects in terms of synchronisations; service object synchronisation, path expression, synchronisation counters
Level 4	Quality-of-service level	Qualification of expected behaviour via quality-of-service parameters (e.g. maximum response delay, quality of the result, expressed as precision)

Table 6: Four Levels of Interface Contracts according to Beugnard (Beugnard et al., 2010; Beugnard et al., 1999)

that goes beyond the syntactical level 1 shall be introduced and analysed, namely Franca IDL. This IDL is used in practice and is a textual language for specification of APIs. While Franca is predominantly utilised by automakers and suppliers in the automotive industry like BMW, Bosch and Continental, it also finds application in non-automotive domains (Birken, 2017). The GENIVI consortium relies on Franca as an IDL and integration tool to provide an open platform for developing In-Vehicle Infotainment systems. A specification using Franca IDL offers the following aspects:

- data types, constants and initialiser expressions
- actual interfaces consisting of attributes, methods and broadcasts
- protocol state machines for specification of dynamic behaviour

Thus, it incorporates characteristics of level 3 of Beugnards classification scheme.

Use Case IDLs, such as Franca IDL, have diverse applications beyond traditional use cases, including documentation and communication. IDLs facilitate code generation, enabling automated testing and verification, which can save time and reduce errors. Additionally, IDLs aid in the development process by generating code that implements the specified interface. Test interfaces can be defined using IDLs to simplify isolated software component testing and facilitate issue identification and isolation during testing. IDLs foster interoperability by enabling communication between software components written in different programming languages through a standardised interface. This standardisation promotes streamlined integration of software components from different vendors. Contract-based interface specifications define the assumptions and guarantees of software components,

instilling client confidence in using the component. Automated testing and verification using contract-based interface specifications enhances the system's reliability and safety.

Features Based on the classification scheme, IDLs categorised as level 1 mainly focus on the static elements of interfaces, including data types, attributes and operations. However, there are often constraints on the order of interactions on the interface, such as requiring the *init()* method to be called before any other interaction. While such dynamic behaviors can be specified in the textual documentation of the interface, Franca IDL goes beyond level 1 by offering protocol state machines (PSMs) that can be defined for each interface. PSMs allow for the expressive specification of the exact order of events that any implementation of the interface must support, with any sequence not covered by the PSM considered a contract violation. PSMs can ease the identification of hard-to-find integration errors by defining the procedures and temporal conditions at the interface. Additionally, the number of permissible invocation sequences can be significantly reduced, simplifying the testing of specified behavior or monitoring during runtime (itemis AG, 2018).

Pitfalls Franca IDL is specifically used to integrate software components. When thinking of systems engineering or even Systems of Systems engineering, one would also need these abilities of contracts within tools that go beyond software components. Furthermore, it might be beneficial to have the possibility of including additional characteristics of Beugnards interface classification scheme, e.g. by having the possibility to define and access behavioural contracts in the form of pre- and post-conditions or by incorporating non-functional properties like signal qualities. Hugues and Procter claim to extend interface contracts from software engineering to systems design e.g. via the Architecture Analysis and Design Language (AADL) (Hugues & Procter, 2022). Nevertheless, according to Benveniste, the *"use of contract-based techniques in system design is in its infancy in industry"* (Benveniste et al., 2018). In a similar vein, in 2022, Hugues recommended the wider use of the concept of contracts to facilitate the development of software-intensive systems. It is therefore necessary to develop concepts like Franca Interface Definition Language (IDL) and introduce them to the industry to enhance contract-based interface descriptions, as per Hugues' suggestion (Hugues & Procter, 2022).

4.2. Analysis of Expert Interviews

In the following, the core statements of the expert interviews shall be elaborated. It is not possible to delve into every nuance of the experts' statements due to the limited extent of this thesis. Hence, in order to summarise the expert interview on a qualitative basis, the central statements were grouped into categories in accordance with the initial research

questions RQ1 to RQ3. In the following paragraphs, the analysis will make references to the specific interviews. Please consult the appendix and use the following references: for ID1, see A.2.3.1; for ID2, see A.2.3.2; for ID3, see A.2.3.3; for ID4, see A.2.3.4; and for ID5, see A.2.3.5.

4.2.1. General Statements

Firstly, some generic statements shall be summarised as these give insight into the current view on interfaces in practice and help in understanding the interviewees' perspectives.

Interface Definition The interviewees give different definitions of what an interface is according to their understanding and viewpoint. Here, interfaces are denoted as *"barriers"* (refer to A.2.3.2), as well as *"touch-points of two objects"* (refer to A.2.3.1). In this context, a touch-point is regarded as a human also while the human interface is seen as most complex interface. Here, language or communication are specifically mentioned. A further definition highlights the flow of information or material as it defines interfaces as the *"connection between two system elements: How are they connected and how can they exchange through this?"* (refer to A.2.3.3).The interface definition of expert ID4 goes into a similar direction by defining interfaces as an *"interaction between two components"* (refer to A.2.3.4). ID5 extends this by stating that in his understanding an interface is the point where *"one system ends and the other one begins"* (refer to A.2.3.5).

Interface Characteristics Interfaces are regarded as the *"stepmothers of systems engineering"* by interviewee ID3. He questions why the majority of practitioners disregards interfaces. Furthermore, he mentions that a crucial point of interfaces is their silence. As *"interfaces don't talk"* (refer to A.2.3.3), one might put two systems together via their interfaces but cannot be sure that these systems interoperate. One might philosophise here that "talking" interfaces could improve situations in systems engineering or SoSE. Possibly, this food for thought could be followed in further research.
In contrast to these more negative connoted characteristics, ID5 emphasises positive characteristics of interfaces by arguing that well-defined interfaces can simplify an engineers life.

Interface Classification All interviewees had difficulties to classify the interfaces they face in their projects or working environments. Thus, they all listed different interfaces they are confronted with, having few categorisation or structure in their explanation. Interfaces they mentioned were in summary:

- ID1: customer interfaces, process interfaces, work-product-interfaces, hardware-software-interfaces (interdisciplinary interfaces)
- ID2: organisational and functional interfaces, process interfaces, SW interfaces, interface between monitoring and function level, interface between control units
- ID3: network interfaces between control units, radio data transmission, optical interfaces, GPS, mechanical interfaces, hydraulic interfaces, human-machine interface
- ID4: customer interfaces, software-software interfaces, hardware-software interfaces
- ID5: mechanical interfaces (e.g. spatial, pneumatic, hydraulic), electrical interfaces, interfaces to control technology, interfaces related to flow: material flow

The interfaces which the interviewees described serve as an understanding of their viewpoint and help in understanding their statements. Interestingly, these lists vary between the interviewees. This might be justified by their incorporation of different roles and domains. ID1 additionally added that he perceives the topic as very multilayered and sees a lot of dependencies between interfaces. The expert misses a generally agreed-upon overview of interface classifications.

4.2.2. Usage Scenarios

Use Case Dependability Each of the five interviewees affirmed that their approach to interface specification depends on their specific use case. There is no existing general scheme or pattern that maps interface specification approaches to use cases and the selection of an approach is typically flexible and informal. ID3 even noted some *"confusion"* in this regard (see A.2.3.3). Additionally, the level of detail described in interface specification varies among the experts depending on the system's level of decomposition.

Life Cycle Since all the interviewees are resident in the development phase of the system life cycle, their use cases for interface specifications primarily focused on this phase. While a few use cases were mentioned for the production, operation/service and recycling/disposal phases, this should not be interpreted as a lack of use cases for those phases.
Within the development cycle, interface specifications are needed along the complete V-model according to the experts. Conforming to their work area, ID1 specifically mentions the need within the left wing of the V-model, while ID2 mentions its right wing. Furthermore, especially the beginning of the projects is emphasised by interviewees ID1, ID3 and ID4. All three highlight that it is crucial to have interface definitions at the beginning of projects to have a common understanding of the system. According to ID3, for mechanical aspects this might be harder to achieve as mechanical dimensions might not be clear from the beginning. For electrical parts on the other hand, the communication

structures are available early in a project which makes interface specifications creation easier in this respect. ID5 builds upon this rationale by asserting that the mechanical interfaces initially begin with a rough outline at the outset of a project and are progressively defined with greater precision as the project progresses.

Use Cases The use cases described by the experts correspond to each other to a large extent. It was mainly mentioned to use interface specifications for communication reasons. Here, ID1 refers to communication to customers, to communication between internal development departments and to communication between the different disciplines of a system. ID2 argues in a similar way by mentioning the information exchange between software departments, project departments and the respective customer. ID3 goes into the same direction and additionally highlights the different development locations of systems which require clear interface specifications. The content of the interface specification is regarded as an *"information exchange format"* (refer to A.2.3.2) and the *"contentual input vector"* (refer to A.2.3.3) for all development partners.
Further use cases that are explicitly mentioned are:

- coordination and harmonisation between disciplines and development steps, called *"mediator"* (ID1)
- translation of units (ID1, ID2)
- development across locations, thus to decouple engineering, to achieve concurrent engineering, to decompose system into sub-problems (ID1, ID3)
- to provide rules regarding roles and responsibilities (ID4)
- debugging (ID2)
- to achieve comprehension models and describe systems (ID3)
- to clearly define and delineate tasks (ID4, ID5)
- to decouple engineering (ID1, ID5)
- to check interactions (ID5)

4.2.3. Interface Specification Approaches

Process ID3 outlined a process for creating interface specifications, which involves mapping requirements to architectures to make interfaces visible, and then recording these interfaces as derived requirements. These derived requirements can be utilised for integration and testing purposes. One takeaway from the interviews is the need for additional examinations regarding the interface identification process. Incorporating these questions in future expert consultations could be beneficial.

Specification Approaches In a nutshell, the interface specification approaches mentioned by the experts were document-based as well as model-based approaches. ID1, ID3, ID4 and ID5 use mostly document-based approaches in natural language. Here, HSI as well as DIA (please refer to the previous chapters) were explicitly mentioned by ID1. Especially the DIA was seen as a showpiece for a good interface specification within its specific use case by ID1: *"For me, the DIA as an interface specification approach in the individual area of application is very mature and doesn't need any improvement"* (refer to A.2.3.1). This statement clearly shows, that the interface specification approach is highly dependent by its specific use case. Although following model-based approaches within the development, ID3 confirms to use textual specifications for interfaces only. Within these documents, e.g. Ethernet-based descriptions or CAN-matrices are used. According to ID1 and ID3, the document-based approaches are created manually as their tools don't provide an automated generation of interface specifications.
ID1 and ID2 in addition to textual approaches use semi-formal methods, e.g. by utilising SysML. ID2 uses a complete model-based approach and emphasises the ability of reuse within these models. Additionally, the possibility of versioning as well as unit conversion in the model-based approach are mentioned as advantages.
Moreover, ID5 employs model-based methods, utilising computer-aided design (CAD) tools in both two-dimensional and three-dimensional formats, in the relevant industry domain. Notably, ID5 emphasises the need for interfaces to be discussed in written or spoken form, in addition to the model-based approaches. Any changes made to these interfaces must be clearly indicated within the models themselves.

Standards Different standards exist according to all of the experts. One of these standards are established company internal templates for interface specifications, as used for DIA or HSI. The DIA for example as a tabular interface specification is quite stable in its version and is only updated minorly. It has a fixed structure and is only filled with the relevant content. Equally, the HSI is mostly based on the ISO26262 template (refer to ID1).
Additionally, standardised units are used within model-based approaches to achieve consistency and to prevent error sources with regard to unit conversions (refer to ID2).
Further standards that are used mostly appear at external interfaces, e.g. at the communication of an agricultural machine with external systems. This is realised via CAN-interfaces or ISO-Bus interfaces. Internally, only few standards exist according to ID3. Here, standards mostly occur via implicit parameters. For the description of a signals' quality, so-called ASIL or AgPL levels are defined which allow to conclude on the signals' qualities implicitly.

The semantic is e.g standardised by a company-internal language handbook in the agricultural domain. This handbook defines a normalised English which ensures a company-wide meaning of words in different languages. Standards are mentioned by the experts ID1, ID2 and ID3 to generally simplify the development, especially in the context of safety or security development.

Interface Contracts Interface contracts had to be explained to ID1 and ID2 shortly. This missing knowledge shows their significance within the current practice. Having explained the concept of interface contracts, both agreed on interface contracts' advantages but also mentioned that these are not used within their working environment. The same holds for ID4. ID3 mentioned to only use interface contracts implicitly, meaning these contracts are implicitly available in the engineers minds but are not written down in an explicit format. Nevertheless, the expert underlines the advantages of interface contracts and its promising abilities in systems engineering as well as Systems of Systems engineering.
ID5 explains interface contract methods employed in his field of work, although his descriptions primarily pertain to contracts implemented within systems, rather than at the interfaces between systems. Nonetheless, fundamental methods for interface contracts, which are largely implicit in nature, are commonly utilised within this domain.

4.2.4. Guidelines and Improvement Potential

General All interviewed experts see improvement potentials with regard to interface specifications. Specifically, the increased need of interface specifications in processes is mentioned. Here, the interface specifications shall ease and foster reuse and information hiding (refer to ID1). Generally, interfaces shall be addressed in a more abstract way to ease their specifications (refer to ID3). ID4 has identified an opportunity for improvement, which is the frequent exchange of experts who specialise in these topics.

"Good Specifications" The experts provided a lengthy list of attributes that define a "good" interface specification. To provide a summary, the following characteristics or properties were mentioned:

- easy to read
- clarity
- usage of strong words
- visualisation of coherences
- description of active and passive states
- inclusion of signal qualities

- low computing times (in model-based approaches)
- distinct description
- umambigous interpretation, no room for interpretation
- make the implicit explicit
- should possess comprehensive knowledge of both sides of the interface, in order to avoid any potential misunderstandings when interpreting requests
- comprehensible for different stakeholders

4.2.5. Current Challenges

The interviewed experts have identified various challenges with the current interface specification approaches, indicating that they are not yet at their optimal level. ID3 proposes that one possible reason for the current lack of emphasis on interfaces is the relatively small number of systems produced in his domain. According to ID3, losses associated with interface specifications in his domain may be comparatively small when compared to other domains such as automotive production, which involves much larger numbers of units. The author notes that in the automotive domain, there are large (interface) standards such as AUTOSAR.

Tooling A crucial point that was mentioned by different experts was the available tooling used for interface specifications. Here, especially ID3 elaborated that unless requirements and architectural tools exist, nowadays one is not able to generate interface specifications from the architectural descriptions in an automated way. According to him, the *"last bit of tooling"* (refer to A.2.3.3) is missing at this point. An advantage would be to describe interfaces directly within modeling tools so that they can be generated out of these with little effort. Furthermore, due to heterogeneous structures within companies, the distribution and penetration of existing tools is missing. This leads to a lacking maturity of the content described in tools and most communication related to interfaces is thus absorbed by human communication.

Human Factor The human factor is a challenge that was repeatedly associated with the creation of interface specifications. Especially at interface specifications that combine different disciplines, this human factor was regarded as essential. According to ID1, the creator of such an interface specifications needs to have different skills combined in one person. These include e.g. a deep understanding of the system itself as well as of the software and hardware parts and important soft skills like open communication or being a mediator between different stakeholders to agree on stable interface specifications (refer to ID1). ID3 extends this reasoning by mentioning that both, *"[...] olympia athletes as*

well as people on a village level exist within a company, therebetween lays the area of tension" (refer to A.2.3.3). Thus, it is a complex task to align these ways of working. Additionally, the expert mentions an example of how different locations within a company may have varying needs. Even if the headquarter of a company with high engineering know-how could create interface specifications in an automated way, it does not mean at the same time, that the rest of the company can read and work with these. Nevertheless, ID3 acknowledges that current interface specifications still have room for improvement, and as a result, much of the coordination between systems relies on the knowledge and communication skills of humans who are familiar with these systems. Thus, all involved persons need to have the same understanding. This is analogously confirmed by ID1 and ID5.

Consistency Another challenge mentioned by several experts is the aspect of consistency of interface specifications. Here, the consistency of units within models is given as an example by ID2. Additionally, according to ID1 and ID3, interface specifications need to be kept consistent over time as well as over the development partners. As at an interface always at least two systems are engaged, this is a crucial point and a challenge at the same time (refer to ID1, ID2, ID3).

Semantics The semantics element is a challenge that ID4 mentions in particular. So, unless information can be transmitted from one interacting component, be it a human, a component or any kind of system in the broad sense, the meaning of the transmitted information needs to be understood and interpreted correctly. It therefore is not sufficient to overcome the hurdle of transmitting information, but the information above this needs to be correctly understood.

5. Solution Design

This chapter synthesises the findings from the previous analyses in order to establish a foundation for developing a customised approach to interface specification.

5.1. Consolidation of Analysis Results

5.1.1. Overview on Characteristics of Interface Specifications

Based on the previously analysed interface specification approaches and the insights from the expert interviews, the approaches identified can be categorised into the five main characteristics: *Content*, *Appearance*, *Formality*, *Level of Specification* and *Description*. Every interface specification approach found includes these characteristics while different combinations of these characteristics exist. Figure 27 visualises this overview of the existing characteristics of interface specifications. The model is based on UML to provide an overview that speaks for itself.

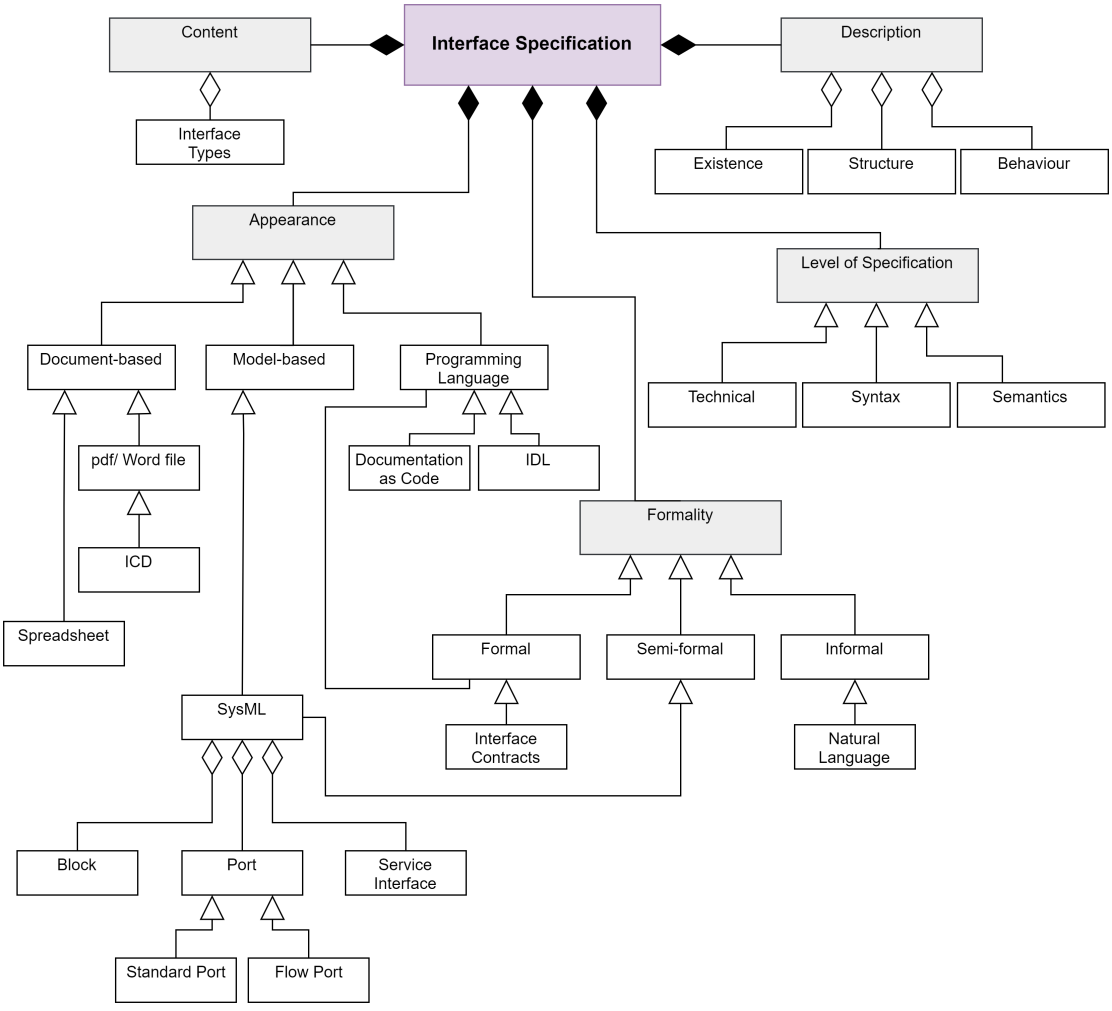


Figure 27: Overview on Characteristics of Interface Specifications

As illustrated in the figure, each of the essential characteristics can be dissected into smaller components. In this chapter, the properties of these characteristics shall be extracted and analysed to deconstruct the primary problem into more manageable sub-problems.

5.1.2. Overview of Use Cases

In addition to the overview of interface specification approaches, the overview of interface specifications' use cases is depicted in Figure 28 aligning to RQ2. Here, only the development phase of the systems life cycle is depicted to focus on one specific phase and as most approaches analysed originate from the development phase. Also, the interviewed experts evaluated mostly only the development phase as they where working within this phase. The use case overview from Chapter 3 was extended by the use cases repeatedly found during analysis.

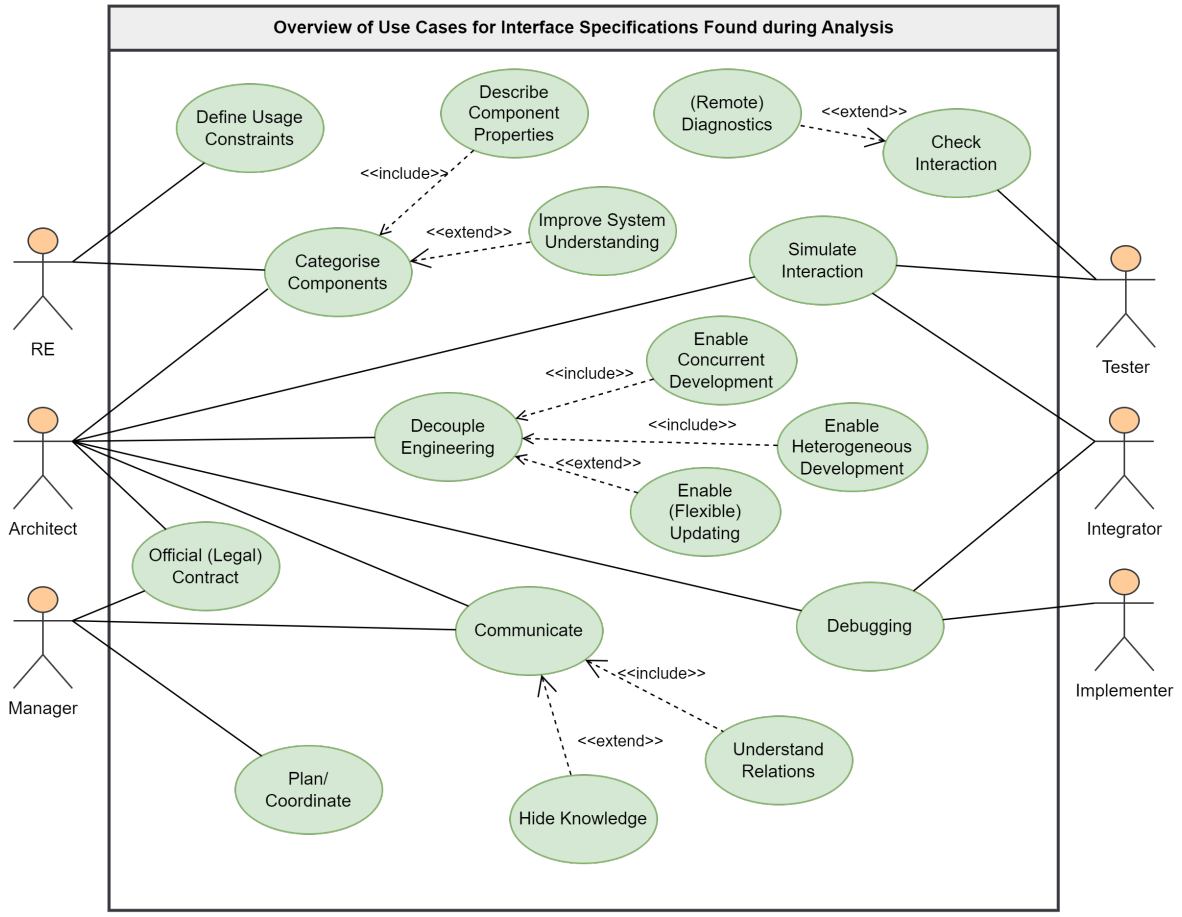


Figure 28: Use Cases for Interface Specifications Found during Analysis

The most significant and frequently mentioned use cases identified in the analysis conducted will be further elaborated. These include the following five use cases: *communicate*, *debugging*, *decouple engineering*, *define usage constraints* and *official (legal) contract*. In

Table 7, a comprehensive overview of the use cases is presented alongside their corresponding visual representation and detailed description.

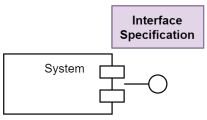
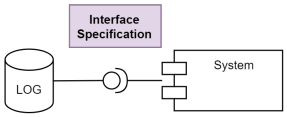
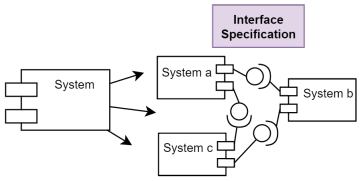
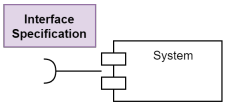
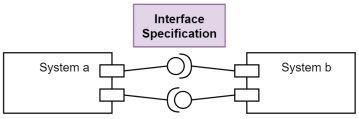
NAME	VISUALISATION	DESCRIPTION
Communicate		The interface specification serves as a means to communicate and explain the system to various stakeholders. It enables stakeholders to comprehend the system's relationships. By only conveying the system's function through the interfaces or their interface specification, the knowledge of the system can be concealed in a black-box manner.
Debugging		The interface specification is used to debug systems. It is thus used to detect issues or errors and to resolve these.
Decouple Engineering		The interface specification is used to decouple engineering in order to concurrently engineer a system. It thus is used as the input vector of development teams. Via the interface specification, reuse, heterogeneous development or flexible updating of systems can be enabled.
Define Usage Constraints		The interface specification is used to constrain the systems usage. It can thus limit the input that the system is allowed to receive.
Official (Legal) Contract		The interface specification serves as a (legal) contract between two systems. It can thus be used to manage responsibilities among others.

Table 7: Overview of Selected Use Cases

5.2. Extraction of Features

To extract the features of interface specification approaches, an analysis of their characteristics is required. This analysis will involve examining the peculiar features and pitfalls of the characteristics, based on the results of the overall analysis and the expert interviews. The analyses of the five key characteristics content, appearance, level of specification, description and formality will be presented in tabular formats. The tables include columns outlining the key features or benefits of each characteristic, their drawbacks and significant

examples.

5.2.1. Content

The characteristic *Content* is a special case among the five characteristics that were identified. Allocating features or pitfalls for this characteristic is not straightforward, unlike for the remaining characteristics. The content of an interface specification depends on the interface types used, the domain being followed and their respective use cases. Therefore, a model of this characteristic is presented in Figure 29 to provide guidance on which content to select for interface specification approaches. The classification of interface types identified earlier (refer to Chapter 2) serves as a basis. Based on the analysis of approaches, the distinction between internal/external and technical/organisational interface types, as well as the classification of spatial, energy, information and material types stated by Pimmler and Eppinger (Pimmler & Eppinger, 1994), will be used as the foundation for distinguishing between the contents of interface specifications. This classification was chosen because the analysis of different approaches showed that approaches differ mainly between technical or organisational interface descriptions. Additionally, the content or information covered within a specification differs depending on whether it is distributed within a company or shared with external parties. Finally, the interface description differs depending on the actual content that is shared between two systems via the interface.

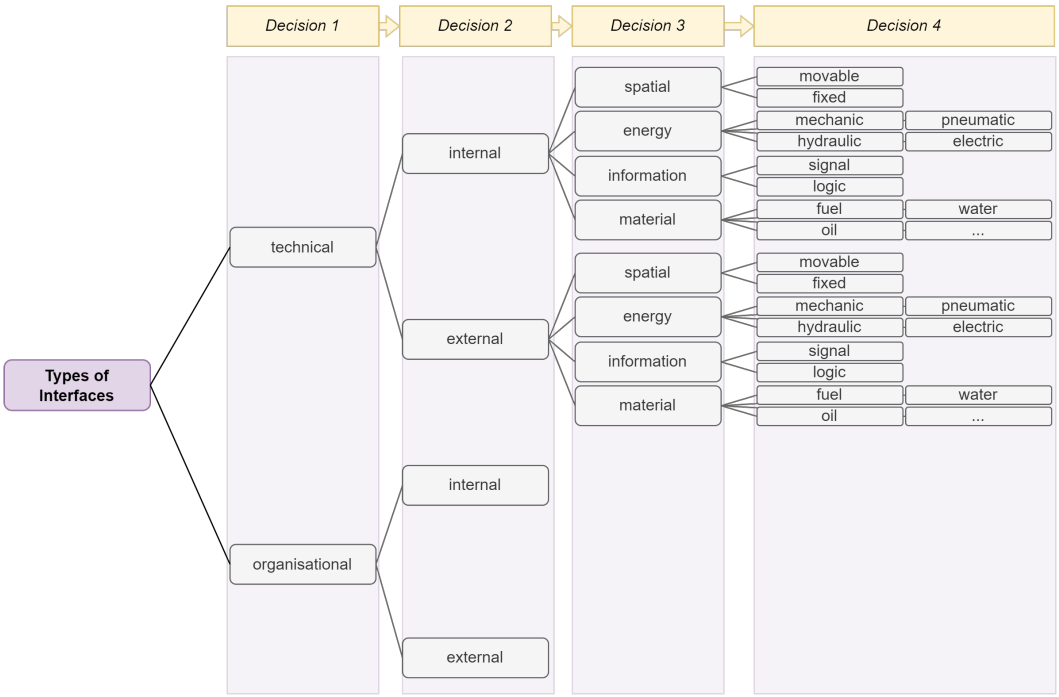


Figure 29: Decision Tree for the Content of Interface Specifications

The differentiation of Pimmler and Eppinger's categories, namely spatial, energy, informa-

tion and material, can be further refined. For instance, the aspect of energy can be specified more precisely by indicating whether it is mechanical, hydraulic, electrical or pneumatic energy. This refinement is based, among other works, on Rahmanis' research (Rahmani & Thomson, 2011). Ultimately, the content presented in the interface specification depends on the use case and the information required by the stakeholder. The overview presented serves as an initial guideline and aid for classification. As a supplementary note, it is important to not only take into account the intended interfaces of a system, but also to consider unintended interfaces within the system, as Davies (Davies, 2020) points out. Unintended interfaces can be caused by external factors such as noise or vibration to give two examples.

5.2.2. Appearance

The characteristic *Appearance* can be subdivided into three categories that were identified in the context of interface specifications. These include document-based approaches such as spreadsheets or PDF/Word files, model-based approaches and program-based approaches. The features, drawbacks and examples of these approaches are presented in Table 8.

CHARACTERISTIC	FEATURES/BENEFITS	DRAWBACKS	EXAMPLES
Document-based	Flexible and effective. Straightforward. No special tooling needed. Widely understood. Readers/ stakeholders don't need special background.	No actual link between the corresponding system data. Manual tracking and reflection of changes. Multitude of documents in complex systems. Difficult verification of completeness/ consistency. Difficult version control.	ICDs (e.g. USB, GPS), spreadsheets (e.g. DIA, HSI)
Model-based	Model as "single source of truth". Enforces consistency. Can be a basis for generating ICDs. Easy tracking of changes. Traceability (e.g. link to requirements). Manages complexity. Enabler for collaboration. Reduction of ambiguities. Eases visual and electronic inspection. Version control system. Concurrent access by team members.	Special tooling needed (including licensing and knowledge). Difficult to launch in heterogeneous companies or domains. Rules and (design) guidelines need to be followed.	SysML, e.g. MBSE in spacecraft
Programming Language	Use of conditions. Component verification (black-box testing). Code generation.	Semantic information possibly only via comment mechanism. Special tooling needed (including licensing and knowledge). Difficult to launch in heterogeneous companies or domains.	IDLs, e.g. CORBA IDL, Franca IDL, OMG IDL

Table 8: Overview of Approaches Depending on their Appearance

5.2.3. Level of Specification

The *Level of Specification* of an interface specification is based on the interoperability levels that were already discussed in Chapter 3. It thus shall be understood as the classification

into the three categories technical, syntax and semantics. Further details can be found in Table 9.

CHARACTERISTIC	FEATURES/BENEFITS	DRAWBACKS	EXAMPLES
Technical	Clear understanding of communication protocols and data formats needed for integration. Ensures communication of components. Improves system performance through data transmission.	Specialised technical knowledge to implement and troubleshoot.	Definition of communication protocol, e.g. CAN-protocol
Syntax	Definition of structure and format of transferred data. Communication without needing to know the details of how internals work.	Dependency on structures. Misinterpretation possible.	Specification of names and types of parameters that are passed
Semantics	Same mutual understanding. Unambiguity. Allows reuse in different contexts.	Initial creation effort. Maintenance effort. Costs.	Glossaries or dictionaries, e.g. ECL@SS standard

Table 9: Overview of Approaches Depending on their Level of Specification

5.2.4. Description

Another identified characteristic is the (level of) *Description*, which can be categorised into three levels of granularity.

CHARACTERISTIC	FEATURES/BENEFITS	DRAWBACKS	EXAMPLES
Existence	Low effort. Flexibility. Simple understanding.	Few information on interface available. Ambiguity. Lack of clarity. Focus on static aspects. Limited reusability. Potential of misinterpretation.	Port in MBSE- Internal Block Diagram
Structure	More information available. Easier integration. Decoupling from integration.	Focus on static aspects (no temporal aspects). Limited testing possible. Possibly missing information. Possibly poor interoperability and misuse.	Description of diameter and drill depth for drilled holes (as spatial interfaces)
Behaviour	Can show temporal aspects. Testing possible. Formal verification. Can provide guarantees. Shows dynamic aspects. Clarity. Can show quality of expected behaviour.	High effort. Dependencies. Possibly over-specification. Inflexibility. Higher maintenance efforts.	Protocol State Machine connected to an interface

Table 10: Overview of Approaches Depending on their Description

At the simplest level, only the existence of an interface is described without any further information. The second level involves the description of the interface's structure, which may include mentioning the information or material that is exchanged or the general composition of the interface. The third level of interface description involves describing its behaviour, which includes considering not only the behaviour of operations but also the behaviour of objects in terms of synchronisations, as well as possibly the expected

quality of behaviour (e.g. in form of non-functional requirements). Further details on this differentiation are provided in Table 10.

5.2.5. Formality

The *Formality* level of interface specifications can be divided into three categories: informal, semi-formal and formal. More information on these categories can be found in Table 11.

CHARACTERISTIC	FEATURES/BENEFITS	DRAWBACKS	EXAMPLES
Informal	Well known. Easy use. Easy to create. No special tooling needed. Suitable for small-scale projects. Requires minimal technical knowledge, suited for communication between technical and non-technical stakeholders.	Lack of precision or clarity (ambiguity). Not machine-readable, limited tool-support. No automated testing or verification. Manual processes. Lack of scalability (consistency and coherence in large-scale projects). Lack of uniformity. Hard to maintain.	Natural language, e.g. English
Semi-formal	Transparency. Linkage between development artefacts. Partial automation possible/ partially machine-readable. Detection of inconsistencies. More comprehensible than formal notations.	Tools and skilled developers needed. Rules or guidelines need to be followed. Only partial automation. Limited expressiveness.	Modeling languages, e.g. ICML modeling language
Formal	Explicit description of contractual properties. Automated testing/ verification. High precision. Detection of interface errors early. Machine-readable. Absence of ambiguity. Generation of Code. Confidence that operations will adhere to the expected properties on interfaces.	High costs. Requires skilled developers and expertise. Expression of temporal dependencies. Strict rules and guidelines to follow. Rigidity, inflexibility.	Contract-based methods, e.g. Design by Contract

Table 11: Overview of Approaches Depending on their Formality

5.2.6. Main Features Found

Out of the extraction of features from the five interface specification characteristics, several main features can be extracted. These are displayed in Table 12 with a description to have a common understand of these features for the further reasoning.

5.3. Customised Interface Specification

A key finding from the expert interviews is the high degree of dependence of interface specification approaches on their particular use case. The experts note that there is currently no established framework for identifying an appropriate interface specification approach for a given use case. Therefore, based on an analysis of various interface specification approaches, including their features and use cases, a customised approach is presented in this study. This approach involves mapping the characteristics of identified interface specification approaches to their distinct features and then creating a tailored

FEATURE	DESCRIPTION
Code Generation	The interface specification can be used to generate code.
Concurrency	Concurrent development and access. The interface specification allows to separate systems into concurrent development teams. To this end, the interface specification captures the needs of all (interdisciplinary) development teams completely and in a wide-scale comprehensive way.
Interdisciplinarity	The interface specification can facilitate the connection or integration of interdisciplinary interfaces, encompassing diverse disciplines such as mechanics, electronics and software.
Interface Contracts	Interface specifications can incorporate interface contracts, which establish rules and requirements governing communication, data exchange and functionality. These contracts ensure that both parties follow the specified standards, promoting proper integration and interoperability.
Legacy Systems	The interface specification can readily incorporate specifications from legacy systems or interface specifications provided by external parties, such as other companies.
(Legal) Contract	The interface specification can serve as a legally binding contract, provided that it is unambiguous and clearly understood by all parties involved. While the use of Smart Contracts is not included in this feature, it may be incorporated within the feature of *Interface Contracts*.
Low Costs	The creation and maintenance of the interface specification is not expensive, resulting in a low level of effort required for its development and upkeep."
Manage Complexity	The interface specification can reflect interfaces of complex and multidisciplinary systems. At the same time, the interface specification provides a clear overview of the complex structures.
Modifiable	The interface specification is easily modifiable and any modifications are reflected in a versioning system, which makes the changes transparent to the reader. As a result, the interface specification can adapt flexibly to any changes.
Reduce Ambiguities	The interface specification can reduce ambiguities.
Testable	The interfaces on which the interface specification is based can be verified and validated using appropriate tools or methodologies. Testing can be automated, making the interface specification machine-readable and facilitating formal verification.
Understanding	The interface specification can be broadly understood. It is widely understandable by several different stakeholders and can be read without the need of special tooling or backgrounds.

Table 12: Description of Main Features

interface specification approach that meets the specific needs of the use case and the required features. The study incorporates results from expert interviews and industry analyses presented in Chapter 4.

5.3.1. Explanation of the Implementation

The customised interface specification approach was explored in this master's thesis by testing various solution alternatives. The conceptual model presented in Chapter 3 was utilised as an input. Initially, it was considered to include all six dimensions of the interface classification scheme in a mapping. However, this approach would have resulted in a complex, multi-dimensional tool. Therefore, the subsequent approach was developed that is universally understandable and possibly more helpful in the context of interface specifications. This approach could be further detailed in future research.
The result of the customised interface specification consists of two stages of two-dimensional mappings. The first stage maps use cases to their required features, while the second stage maps features to interface specification approach characteristics. Figure 30 illustrates

how to use this approach to create a customised interface specification. Specifically, the initial step is to select the use case of interest in the first mapping (1), which leads to the selection of necessary features (2). These features are then used in the second mapping (3) to identify the required characteristics of the interface specification approach (4). With this information, a customised interface specification can be created. An example or proof of concept of this approach will be provided in the following section.

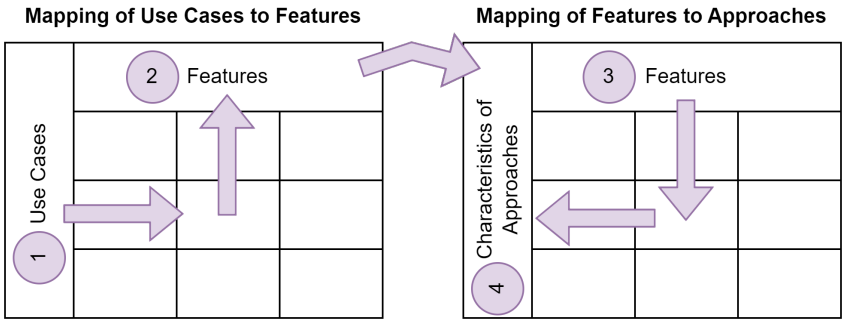


Figure 30: Explanation of the Customised Interface Specification

5.3.2. Mapping: Use Cases to Features

The analysis of interface specifications, along with the expert interviews identified five common use cases that were explained in detail before. To illustrate their required features, the use cases are mapped to the corresponding features in Table 13. The table indicates the varying degree of importance of each feature for each use case by using filled, half-filled and empty circles that translate to *high need*, *medium need* and *low need*. These assessments of degrees were determined based on the analyses that have been performed before.
The user should read the table from left to right, selecting one use case at a time to identify its required or less required features. For instance, the use case *communicate* places high importance on features like *interdisciplinarity*, *manage complexity*, *reduce ambiguities* and *understanding*, while features such as *legacy system* or *testable* are less required.

5.3.3. Mapping: Features to Characteristics of Approaches

Once the essential features have been identified, they must be used as input for the second mapping, which maps the features to the corresponding characteristics of approaches. In this regard, Table 14 shall be consulted, reading from top to bottom resp. left to identify the determined features. Like the previous mapping, the mapping in question includes three varying levels that are represented by circles, with each circle corresponding to a level of accordance: *high accordance*, *medium accordance* and *low accordance*. In cases where there is no correlation between a feature and an approach characteristic, this is indicated

USE CASE	CODE GENERATION	CONCURRENCY	INTERDISCIPLINARITY	INTERFACE CONTRACTS	LEGACY SYSTEMS	(LEGAL) CONTRACT	LOW COSTS?	MANAGE COMPLEXITY	MODIFIABLE	REDUCE AMBIGUITIES	TESTABLE	UNDERSTANDING
Communicate	○	○	●	○	○	○	◐	●	◐	●	○	●
Debugging	●	◐	○	◐	○	○	◐	○	○	◐	●	◐
Decouple Engineering	○	●	◐	◐	●	◐	◐	◐	○	●	◐	●
Define Usage Constraints	◐	◐	◐	●	○	◐	○	◐	◐	◐	●	◐
Official Contract	○	◐	◐	○	◐	●	◐	◐	◐	●	○	●

● = high need ◐ = medium need ○ = low need

Table 13: Mapping of Use Cases to Features

by a hyphen. The assessment is also based on the by analyses that were conducted during previous actions. From the characteristics identified for the interface specifications, the one that best matches the required features is chosen. Consequently, a customised approach

CHARACTERISTIC	CODE GENERATION	CONCURRENCY	INTERDISCIPLINARITY	INTERFACE CONTRACTS	LEGACY SYSTEMS	(LEGAL) CONTRACT	LOW COSTS	MANAGE COMPLEXITY	MODIFIABLE	REDUCE AMBIGUITIES	TESTABLE	UNDERSTANDING
Appearance												
Document-based	○	◐	◐	○	◐	●	●	○	○	-	○	◐
Model-based	◐	●	◐	◐	◐	○	◐	●	●	-	◐	◐
Programming language	●	◐	○	●	○	○	○	◐	◐	-	●	○
Formality												
Informal	○	◐	-	○	◐	◐	●	◐	◐	○	○	●
Semi-formal	◐	●	-	◐	●	●	◐	●	◐	◐	◐	◐
Formal	●	◐	-	●	◐	○	○	●	○	●	●	○
Level of Specification												
Technical	-	◐	○	◐	●	○	●	○	-	○	-	○
Syntax	-	◐	◐	◐	◐	◐	◐	◐	-	◐	-	◐
Semantics	-	●	●	●	○	●	○	●	-	●	-	●
Description												
Existence	-	-	○	○	●	-	●	●	●	○	○	-
Structure	-	-	◐	◐	◐	-	◐	●	●	◐	◐	-
Behaviour	-	-	●	●	◐	-	○	◐	●	●	●	-

● = high accordance ◐ = medium accordance ○ = low accordance - = no influence

Table 14: Mapping of Features to Characteristics of Approaches

for interface specification is obtained. It should be noted that the characteristic *content* is not included in this mapping due to its strong dependence on the interface types employed,

requiring separate consideration. The preceding paragraphs provide guidance on how to address this issue. Ultimately, the outcome of the mapping process is a bespoke interface specification tailored to the specific use case it is intended for.

5.3.4. Proof of Concept

In the following, the implementation of the customised interface specification in form of the two mappings shall be tested and verified with two example cases.

Description of Example Case 1 To test and evaluate the customised interface specification approach developed before, a test iteration with a selected use case will follow. The use case that was selected was *decouple engineering* as this was mentioned within the expert interviews several times, e.g. by interviewee ID1 by stating that *"[...] interface specifications are used to coordinate between the different development teams"*. To this end, in a first step the mapping of use cases to feature needs is used. The respective use case is therefore marked in blue in Table 15. As explained before, the table needs to be read from the left side "use case" to the right side where all features are matched.

USE CASE	CODE GENERATION	CONCURRENCY	INTERDISCIPLINARITY	INTERFACE CONTRACTS	LEGACY SYSTEMS	(LEGAL) CONTRACT	LOW COSTS?	MANAGE COMPLEXITY	MODIFIABLE	REDUCE AMBIGUITIES	TESTABLE	UNDERSTANDING
Communicate	○	○	●	○	○	○	◐	●	◐	●	○	●
Debugging	●	◐	○	◐	○	○	◐	○	○	◐	●	◐
Decouple Engineering	○	●	◐	◐	●	◐	◐	◐	○	●	◐	●
Define Usage Constraints	◐	◐	◐	●	○	◐	○	◐	◐	◐	●	◐
Official Contract	○	◐	◐	○	◐	●	◐	◐	◐	●	○	●

● = high need ◐ = medium need ○ = low need

Table 15: Mapping of Use Cases to Features for Example Case 1

The features that are required mostly, shown by the filled circle, are also marked in blue. These serve as the input criteria for the following mapping of features to approach characteristics. For the selected use case, the mapping reveals the features *concurrency*, *legacy systems*, *reduce ambiguities* and *understanding* as the ones of highest priority. In a next step, these extracted features shall be mapped to needed customised interface specification approach characteristics in Table 16. For this, the table needs to be read starting from the right top, going down to the left side where the characteristics are matched. One starts with the features that were extracted from Table 15. Here, the

CHARACTERISTIC	CODE GENERATION	CONCURRENCY	INTERDISCIPLINARITY	INTERFACE CONTRACTS	LEGACY SYSTEMS	(LEGAL) CONTRACT	LOW COSTS	MANAGE COMPLEXITY	MODIFIABLE	REDUCE AMBIGUITIES	TESTABLE	UNDERSTANDING
Appearance												
Document-based	○	◐	◐	○	◐	●	●	○	○	-	○	◐
Model-based	◐	●	◐	◐	◐	○	◐	●	●	-	◐	◐
Programming language	●	◐	○	●	○	○	○	◐	◐	-	●	○
Formality												
Informal	○	◐	-	○	◐	◐	●	◐	◐	○	○	●
Semi-formal	◐	●	-	◐	●	●	◐	●	◐	◐	◐	◐
Formal	●	◐	-	●	◐	○	○	●	○	●	●	○
Level of Specification												
Technical	-	◐	○	◐	●	○	●	○	-	○	-	○
Syntax	-	◐	◐	◐	◐	◐	◐	◐	-	◐	-	◐
Semantics	-	●	●	●	○	●	○	●	-	●	-	●
Description												
Existence	-	-	○	○	●	-	●	●	●	○	○	-
Structure	-	-	◐	◐	◐	-	◐	●	●	◐	◐	-
Behaviour	-	-	●	●	◐	-	○	◐	●	●	●	-

● = high accordance ◐ = medium accordance ○ = low accordance - = no influence

Table 16: Mapping of Features to Characteristics of Approaches for Example Case 1

approaches with the highest accordances, marked by the filled circles, are considered. In cases where several approaches for one feature could be possible, a second feature is considered to select the characteristic that suits best. For the selected use case *decouple engineering*, the resulting characteristics would be model-based, semi-formal, semantics and behaviour. Thus, the customised interface specification could for instance be a SysML model that describes the systems interfaces behaviour while having a clear semantic. The content or interface types that are described within the interface specification need to be discussed in addition to the found approach. A guideline for their distinction was given before.

In a nutshell, for this specific use case *decouple engineering*, the implementation of the two mappings was able to find a customised interface specification.

Description of Example Case 2 Given that the experts interviewed and the current state of practice did not offer insights on the usage scenarios of Systems of Systems, but instead focused on systems engineering, an additional use case that may be particularly interesting in the context of Systems of Systems engineering will be explained: *define usage constraints*. Based on literature (refer to Chapter 2) and the expert opinions gathered, there is an increasing need to explicitly define the usage constraints of systems, rather than

relying solely on implicit definitions. Building upon the information presented in Table 13, the features that are particularly important for the use case *define usage constraints* include *interface contracts* and *testability*. By incorporating these features into the mapping of approaches to their respective features, Table 17 shows the results highlighted in magenta. For this particular use case, it is advisable to employ a programming language that incorporates semantics and formally defines the behavior of the system, as revealed by this mapping. The use of interface contracts in Systems of Systems aligns with assumptions provided by the customised interface specification, which is supported by both expert opinions and relevant literature (see references of Payne, Bryans, Fitzgerald and Chapter 2).

CHARACTERISTIC	CODE GENERATION	CONCURRENCY	INTERDISCIPLINARITY	INTERFACE CONTRACTS	LEGACY SYSTEMS	(LEGAL) CONTRACT	LOW COSTS	MANAGE COMPLEXITY	MODIFIABLE	REDUCE AMBIGUITIES	TESTABLE	UNDERSTANDING
Appearance												
Document-based	○	◐	◐	○	◐	●	●	○	○	-	○	◐
Model-based	◐	●	◐	◐	◐	○	◐	●	●	-	◐	◐
Programming language	●	◐	○	●	○	○	○	◐	◐	-	●	○
Formality												
Informal	○	◐	-	○	◐	◐	●	◐	◐	○	○	●
Semi-formal	◐	●	-	◐	●	●	◐	●	◐	◐	◐	◐
Formal	●	◐	-	●	◐	○	○	●	○	●	●	○
Level of Specification												
Technical	-	◐	○	◐	●	○	●	○	-	○	-	○
Syntax	-	◐	◐	◐	◐	◐	◐	◐	-	◐	-	◐
Semantics	-	●	●	●	○	●	○	●	-	●	-	●
Description												
Existence	-	-	○	○	●	-	●	●	●	○	○	-
Structure	-	-	◐	◐	◐	-	◐	●	●	◐	◐	-
Behaviour	-	-	●	●	◐	-	○	◐	●	●	●	-

● = high accordance ◐ = medium accordance ○ = low accordance - = no influence

Table 17: Mapping of Features to Characteristics of Approaches for Example Case 2

5.3.5. Evaluation

General Two different scenarios were employed to confirm the validity of the mappings generated. These scenarios were chosen as toy cases and must be extrapolated to real-world scenarios.

The structure to map use cases to their corresponding features initially and then to match these features with appropriate methodologies in a subsequent step creates an interface between these two mappings that could result in information loss.

Expert Consultation To validate the mappings, two out of the five experts were asked to provide their specific opinions on the customised interface specification. ID2 expressed that the use-case driven approach is reasonable and can enhance daily industrial work. He asserted that the mappings can act as an initial guideline for selecting an interface specification approach from the multitude of possibilities by providing a comprehensive overview and direction. Moreover, he recommended that the overviews, mappings and general structure developed in the thesis could serve as a foundation for designing a machine learning model that could be trained with suitable interface specifications. ID5 agreed with the aforementioned reasoning and emphasised the potential of the customised interface specification as a comprehensive overview of current approaches. He advocates the use-case driven approach and suggests that the general methodology could benefit from additional elaboration, tailored to the specific requirements of individual domains.

Room for Improvement The customised interface specification along with its general structure was initially developed as part of this master's thesis and therefore has potential for expansion and enhancement, as also remarked by the two consulted experts. The structure of the customised interface specification allows for easy extension, such as refining use cases in greater detail and adding additional use cases. If the overview becomes too complicated, the mapping could be implemented within a tool. In this context, the attribute *content* could benefit from additional elaboration and it may be worthwhile to incorporate this attribute more closely into the overall mapping and structure.
The purpose of this thesis is to offer a comprehensive overview of interface specifications in a general sense. Nevertheless, if more detailed categorisation is needed, the classification scheme introduced in Chapter 3 can be further used in the customised interface specification approach. By including more of the discovered dimensions, such as different domains or life cycle phases, additional constraints can be taken into account, resulting in a multi-dimensional mapping.

6. Conclusion

6.1. Main Contributions and Findings

The main contribution of this thesis is to shed light on the matter of interface specifications, particularly in the context of increasing connectivity in our lives. Chapter 1 emphasises the need for sound interface descriptions and specifications and this thesis provides a framework for better classifying and approaching such specifications. This work seeks to move away from viewing interfaces as the neglected stepmothers of systems engineering, towards a more intelligent and productive classification and approach to interface specifications.
The analyses conducted in this thesis revealed that a comprehensive overview of interface specifications is currently missing, as is a shared understanding of how to approach them. Additionally, since interface specifications encompass a variety of characteristics, there is no single, optimal solution for specifying them. Therefore, a usage-driven approach in the form of a customised interface specification mapping could assist in identifying an appropriate approach.
Overall, the custom interface specification mappings introduced in this thesis serve a dual purpose: providing situational decision support and facilitating a more thorough examination, classification and evaluation of one's own approaches and procedures.
Regarding the research questions presented at the outset of this thesis, the following section provides a brief overview of the locations where these points were discussed:

RQ1 The literature review in Chapter 2 provides an overview of interface specification approaches as the current state of the art. Furthermore, Chapter 4 analyses the state of practice of interface specification approaches through the review of current approaches and expert consultations. An overview of the characteristics of interface specification approaches identified and analysed is presented in Chapter 5, as depicted in Figure 27.

RQ2 Chapter 3 presents a conceptual model that offers an overview of the use cases of interface specifications, categorised based on the different life cycle phases of a system from their development until their disposal. This overview is further extended through the analysis of practical approaches and expert inputs. In Chapter 5, Figure 28 presents a consolidated view of these use cases, which serves as an input for the customised interface specification approach.

RQ3 Chapter 5 extracts and evaluates the features of the identified interface specification approaches. These features form the basis of the customised interface specification approach presented in Tables 13 and 14.

6.2. Discussion

Although the research carried out on interface specifications involved analysing various approaches and conducting expert interviews, the identified features, pitfalls and use cases might not cover the entire range of possibilities related to interface specifications. The research might represent only a part of this subject area and the author acknowledges the limited access to interface specifications within the Systems of Systems context.
Furthermore, the expert interviews emphasised that interface specifications in general and the proposed customised interface specification approach do not replace the need for intensive collaboration and communication among stakeholders. The human factor is a crucial element in this process, in addition to the interface specification approaches themselves.

6.3. Future Work

As previously mentioned, an initial overview of interface specifications was provided with a specific emphasis on the field of systems engineering. Since an overview of interface specifications was not available in less complex systems engineering, only partial attention could be given to the context of Systems of Systems engineering. While some statements made in the research may be generally applicable and transferable from systems engineering to Systems of Systems engineering, there is no general proof of this validity. Thus, the author suggests that further research, particularly in the Systems of Systems context, is necessary to obtain more detailed investigations and conclusive results.
Furthermore, the customised specification approach introduced in this work can be expanded with additional use cases and more granular features, which could be explored in future research. Additionally, it may be possible to create a layered interface specification based on the proposed mapping, combining different specification approaches in a nested or layered structure. This possibility could be considered in the Systems of Systems context.
Generally, further effort in the context of interfaces as well as within single interface specification approaches could improve the way interfaces are handled and described. Among others, the German association Bitkom has started a standardisation campaign to promote a uniform use of the terms related to interfaces in order to facilitate the substantive understanding in technical exchange on interoperability topics (Reich et al., 2022).
Regarding a special interface specification approach, it is by other means worth noting that the upcoming release of SysML v2 is expected to include enhancements that address interface specification challenges within the model-based context. These enhancements may provide support for managing the increasing number and complexity of interfaces the future holds for us.

References

Abbott, R. J. (1991). On interfaces. *Journal of Systems Integration*, *1*(2), 143–162.

Adler, R., Elberzhager, F., & Siebert, J. (2022). Towards a Roadmap for Trustworthy Dynamic Systems-of-Systems. *arXiv preprint arXiv:2206.06008*.

AUTOSAR. (2022a). Adaptive Platform Release Overview. *Part of AUTOSAR Standard Adaptive Platform, Standard Release R22-11, Document Identification No 782*. Part of AUTOSAR Standard Adaptive Platform, Part of Standard Release R22-11.

AUTOSAR. (2022b). Application Interfaces User Guide. *Part of AUTOSAR Standard Classic Platform, Document Identification No 442*. Part of AUTOSAR Standard Classic Platform, Part of Standard Release R22-11.

Bachmann, F., Bass, L., Clements, P., Garlan, D., & Ivers, J. (2002). *Documenting software architecture: Documenting interfaces* (tech. rep.). Carnegie-Mellon University Pittsburgh, PA Software Engineering Institute.

Batista, T. (2013). Challenges for SoS architecture description. *Proceedings of the First International Workshop on Software Engineering for Systems-of-Systems*, 35–37.

Benveniste, A., Caillaud, B., Nickovic, D., Passerone, R., Raclet, J.-B., Reinkemeier, P., Sangiovanni-Vincentelli, A., Damm, W., Henzinger, T. A., Larsen, K. G., et al. (2018). Contracts for system design. *Foundations and Trends® in Electronic Design Automation*, *12*(2-3), 124–400.

Benzi Lavi, A. T., Avigdor Rosenberg. (2010). System Interfaces -Definition and Design Best Practices. *INCOSE Methodologies Working Group &Tools*.

Beugnard, A., Jézéquel, J.-M., & Plouzeau, N. (2010). Contract Aware Components, 10 years after. *Electronic Proceedings in Theoretical Computer Science*, *37*. https://doi.org/10.4204/EPTCS.37.1

Beugnard, A., Jézéquel, J.-M., Plouzeau, N., & Watkins, D. (1999). Making components contract aware. *Computer*, *32*(7), 38–45.

Bilal, M., Daclin, N., & Chapurlat, V. (2014). System of Systems design verification: problematic, trends and opportunities. In *Enterprise interoperability vi* (pp. 405–415). Springer.

Birken, D. K. (2017). Contract-based software development with Franca. *itemis AG*. https://blogs.itemis.com/en/contract-based-software-development-with-franca

Blyler, J. (2004). Interface management. *IEEE instrumentation & measurement magazine*, *7*(1), 32–37.

BMWi. (2020). Verwaltungsschale in der Praxis - Wie definiere ich Teilmodelle, beispielhafte Teilmodelle und Interaktion zwischen Verwaltungsschalen? (Version 1.0). *Bundesministerium für Wirtschaft und Energie, (BMWi)*.

Boardman, J., & Sauser, B. (2006). System of Systems-the meaning of of. *2006 IEEE/SMC International Conference on System of Systems Engineering*, 6–pp.

Bryans, J., Fitzgerald, J., Payne, R., Miyazawa, A., & Kristensen, K. (2014). SysML contracts for systems of systems. *2014 9th International Conference on System of Systems Engineering (SOSE)*, 73–78.

Bryans, J., Payne, R., Holt, J., & Perry, S. (2013). Semi-formal and formal interface specification for system of systems architecture. *2013 IEEE International Systems Conference (SysCon)*, 612–619.

Burton, A., & Slowik, M. (2023). Five Predictions for Manufacturing. *manufacturing.net*. https://www.manufacturing.net/operations/blog/22724106/five-predictions-for-manufacturing

Cadavid, H., Andrikopoulos, V., & Avgeriou, P. (2022). Documentation-as-Code for Interface Control Document Management in Systems of Systems: A Technical Action Research Study. *European Conference on Software Architecture*, 19–37.

Ceccarelli, A., Bondavalli, A., Froemel, B., Hoeftberger, O., & Kopetz, H. (2016). Basic concepts on systems of systems. In *Cyber-physical systems of systems* (pp. 1–39). Springer.

Conway, M. E. (1968). How do committees invent. *Datamation*, *14*(4), 28–31.

Dahmann, D. J. S. (2015). Systems of Systems Engineering Life Cycle. *Systems of Systems Engineering for NATO Defence Applications (STO-EN-SCI-276)*.

Dahmann, J. S., & Baldwin, K. J. (2008). Understanding the current state of US defense systems of systems and the implications for systems engineering. *2008 2nd Annual IEEE Systems Conference*, 1–7.

Davies, P. (2020). Interface Management–the Neglected Orphan of Systems Engineering. *INCOSE International Symposium*, *30*(1), 747–756.

Eigner, M. (2018). Entwicklung cybertronischer Produkte: Interdisziplinäre Konstruktionsmethoden und -prozesse. https://www.ingenieur.de/fachmedien/konstruktion/produktentwicklung/interdisziplinaere-konstruktionsmethoden-und-prozesse/

Eigner, M., Dickopf, T., & Apostolov, H. (2017). The evolution of the V-model: From VDI 2206 to a System Engineering based Approach for Developing Cybertronic Systems. *IFIP International Conference on Product Lifecycle Management*, 382–393.

Eigner, M., Dickopf, T., & Apostolov, H. (2019). Interdisziplinäre Konstruktionsmethoden und -prozesse zur Entwicklung cybertronischer Produkte – Teil 2/Interdisciplinary Design Methods and Processes to Develop Cybertronic Products – Part 2. *Konstruktion*, *71*, 69–75. https://doi.org/10.37544/0720-5953-2019-01-02-69

Eigner, M., Dickopf, T., Apostolov, H., Schaefer, P., Faißt, K.-G., & Keßler, A. (2014). System lifecycle management: initial approach for a sustainable product development

process based on methods of model based systems engineering. *IFIP International Conference on Product Lifecycle Management*, 287–300.

Embitel Technologies. (2018). Why 'Safety Plan' is Critical in Development of ISO 26262 Complaint Product and Automotive Functional Safety. *Embitel Automotive & IoT Blog*. https://www.embitel.com/blog/embedded-blog/iso-26262-compliant-safety-planning-for-functional-safety-automotive

Embitel Technologies. (2020). Adaptive AUTOSAR vs Classic AUTOSAR: Which Way is the Automotive Industry Leaning? *Embitel Automotive & IoT Blog*. https://www.embitel.com/blog/embedded-blog/adaptive-autosar-vs-classic-autosar

Faldik, O., Payne, R., Fitzgerald, J., & Buhnova, B. (2017). Modelling system of systems interface contract behaviour. *arXiv preprint arXiv:1703.07037*.

Forte, S., Göbel, J. C., & Dickopf, T. (2021). System of systems lifecycle engineering approach integrating smart product and service ecosystems. *Proceedings of the Design Society*, *1*, 2911–2920.

Fosse, E., & Delp, C. L. (2013). Systems engineering interfaces: A model based approach. *2013 IEEE Aerospace Conference*, 1–8.

Frese, T., Hatebur, D., Côté, I., & Heisel, M. (2019). Integration of Development Interface Agreement, Supplier Safety Assessment and Safety Management for Driver Assistance Systems. *Mobilität in Zeiten der Veränderung: Technische und betriebswirtschaftliche Aspekte*, 241–251.

Friedenthal, S., Moore, A., & Steiner, R. (2015). Modeling Structure with Blocks. In S. Friedenthal, A. Moore, & R. Steiner (Eds.), *A practical guide to sysml (third edition), chapter 7* (Third Edition, pp. 115–183). Morgan Kaufmann. https://doi.org/https://doi.org/10.1016/B978-0-12-800202-5.00007-2

Friedenthal, S., & Oster, C. (2017). *Architecting Spacecraft with SysML: A Model-Based Systems Engineering Approach*. CreateSpace Independent Publishing Platform.

Fürst, S., Mössinger, J., Bunzel, S., Weber, T., Kirschke-Biller, F., Heitkämper, P., Kinkelin, G., Nishikawa, K., & Lange, K. (2009). AUTOSAR–A Worldwide Standard is on the Road. *14th International VDI Congress Electronic Systems for Vehicles, Baden-Baden*, *62*, 5.

Gianni, D., D'Ambrogio, A., Simone, P. D., Lisi, M., & Luglio, M. (2012). Model-based Interface Specification for Systems Integration in Systems of Systems Engineering. *INCOSE International Symposium*, *22*(1), 2040–2052.

Gorod, A., Sauser, B., & Boardman, J. (2008). System-of-systems engineering management: A review of modern history and a path forward. *IEEE Systems Journal*, *2*(4), 484–499.

Groen, E. C., Adler, R., Elberzhager, F., Siebert, J., & Liggesmeyer, P. (2022). Anwendungsfälle zu dynamischen Systemen der Systeme der Zukunft.

Guédria, W., Naudet, Y., & Chen, D. (2008). Interoperability maturity models–survey and comparison. *On the Move to Meaningful Internet Systems: OTM 2008 Workshops: OTM Confederated International Workshops and Posters, ADI, AWeSoMe, COMBEK, EI2N, IWSSA, MONET, OnToContent+ QSI, ORM, PerSys, RDDS, SEMELS, and SWWS 2008, Monterrey, Mexico, November 9-14, 2008. Proceedings*, 273–282.

Hattrup-Silberberg, M. (2021). Der ungenutzte Datenschatz: Durch Fahrdaten jährlich mehrere Hundert US-Dollar Wertpotential pro Auto möglich. *McKinsey & Company Pressemitteilung*. https://www.mckinsey.de/news/presse/2021-01-10-car-data

Hause, M. (2018). Systems interface management with MBSE: from theory to modeling to reality. *INCOSE International Symposium*, *28*(1), 1012–1026.

Heinecke, H., Bielefeld, J., Schnelle, K. P., Maldener, N., Fennel, H., Weis, O., Weber, T., Ruh, J., Lundh, L., Sandén, T., Heitkämper, P., Rimkus, R., Leflour, J., Gilberg, A., Virnich, U., Voget, S., Nishikawa, K., Kajio, K., Scharnhorst, T., & Kunkel, B. (2006). AUTOSAR – Current results and preparations for exploitation.

Herzig, S. J., Karban, R., Brack, G., Michaels, S. B., Dekens, F., & Troy, M. (2018). Verifying Interfaces and generating interface control documents for the alignment and phasing subsystem of the Thirty Meter Telescope from a system model in SysML. *Modeling, Systems Engineering, and Project Management for Astronomy VIII*, *10705*, 319–335.

Hugues, J., & Procter, S. (2022). Contracts in System Development: From Multi-Concern Analysis to Assurance with AADL. *IEEE Software*.

ISO. (2011a). ISO International Standard 26262: Road Vehicles - Functional Safety. *ISO 26262-1:2011*.

ISO. (2011b). ISO/IEC/IEEE 29148 International Standard - Systems and software engineering — Life cycle processes — Requirements engineering. *ISO/IEC/IEEE 29148:2011(E)*.

ISO. (2017). ISO/IEC/IEEE 24765 International Standard - Systems and software engineering–Vocabulary. *ISO/IEC/IEEE 24765:2017(E)*, 1–541. https://doi.org/10.1109/IEEESTD.2017.8016712

itemis AG. (2018). Franca User Guide. *Release 0.12.0.1*.

Jamshidi, M. (2008). System of systems engineering-new challenges for the 21st century. *IEEE Aerospace and Electronic Systems Magazine*, *23*(5), 4–19.

Jonkers, H. (2000). Ispec: Towards practical and sound interface specifications. *International Conference on Integrated Formal Methods*, 116–135.

Jonkers, H. (2004). Interface specification: A balancing act. *International Symposium on Component-Based Software Engineering*, 5–6.

Karban, R., Crawford, A. G., Trancho, G., Zamparelli, M., Herzig, S., Gomes, I., Piette, M., & Brower, E. (2018). The OpenSE Cookbook: a practical, recipe based collection of patterns, procedures, and best practices for executable systems engineering for the Thirty Meter Telescope. In G. Z. Angeli & P. Dierickx (Eds.), *Modeling, systems engineering, and project management for astronomy viii* (107050W). SPIE. https://doi.org/10.1117/12.2312281

Karban, R., Hauber, R., & Weilkiens, T. (2009). MBSE in telescope modeling. *INCOSE Insight*, *12*(4), 24–31.

Kitchenham, B., & Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. *2*.

Kozma, D., Varga, P., & Larrinaga, F. (2021). System of systems lifecycle management—a new concept based on process engineering methodologies. *Applied Sciences*, *11*(8), 3386.

Lalli, V. R., Kastner, R. E., & Hartt, H. N. (1997). *Training Manual for Elements of Interface Definition and Control* (tech. rep.).

Liker, J. K. (2021). *Toyota way: 14 management principles from the world's greatest manufacturer*. McGraw-Hill Education.

Louadah, H., Champagne, R., & Labiche, Y. (2014). Towards Automating Interface Control Documents Elaboration and Management. *ACES-MB@ MoDELS*, 26–33.

Louadah, H., Champagne, R., Labiche, Y., & Guéhéneuc, Y.-G. (2016). A data extraction process for avionics systems' interface specifications. *2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, 544–554.

Macher, G., Sporer, H., Armengaud, E., Brenner, E., & Kreiner, C. (2015). Using model-based Development for ISO26262 aligned HSI Definition. *CARS 2015-Critical Automotive applications: Robustness & Safety*.

Macher, G., Sporer, H., Brenner, E., & Kreiner, C. (2018). Signal-Layer Security and Trust-Boundary Identification based on Hardware-Software Interface Definition. *J. Ubiquitous Syst. Pervasive Networks*, *10*(1), 1–9.

Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering: The Journal of the International Council on Systems Engineering*, *1*(4), 267–284.

Möhringer, S., Gausemeier, J., et al. (2002). An Interface Specification for Principle Solutions Supporting the Cross-Domain Design of Mechatronic Systems. *DS 30: Proceedings of DESIGN 2002, the 7th International Design Conference, Dubrovnik*, 533–538.

Muscarella, S., Osaisai, M., & Sheard, S. (2020). Systems and software interface survey. *INCOSE International Symposium*, *30*(1), 1305–1323.

NASA. (2007). Systems engineering handbook. *National Aeronautics and Space Administration*.

Naumann, S., Dick, M., Kern, E., & Johann, T. (2011). The greensoft model: A reference model for green and sustainable software and its engineering. *Sustainable Computing: Informatics and Systems*, *1*(4), 294–304.

Object Management Group. (2018). *Interface definition language version 4.2*. OMG.

Pakala, H. K., Oladipupo, K., Käbisch, S., Diedrich, C., et al. (2021). Integration of asset administration shell and Web of Things. *https://opendata. uni-halle. de//handle/1981185920/41505*.

Parslov, J. F., & Mortensen, N. H. (2015). Interface definitions in literature: A reality check. *Concurrent Engineering*, *23*(3), 183–198.

Payne, R., Bryans, J., Fitzgerald, J., & Riddle, S. (2012). Interface specification for system-of-systems architectures. *2012 7th International Conference on System of Systems Engineering (SoSE)*, 567–572.

Payne, R. J., & Fitzgerald, J. S. (2010). Evaluation of architectural frameworks supporting contract-based specification. *School of Computing Science Technical Report Series*.

Payne, R. J., & Fitzgerald, J. S. (2011). Contract-based interface specification language for functional and non-functional properties. *School of Computing Science Technical Report Series*.

Pimmler, T. U., & Eppinger, S. D. (1994). Integration analysis of product decompositions. *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, *12822*, 343–351.

Plattform Industrie 4.0. (2022). Details of the asset administration shell. *Federal Ministry for Economic Affairsand Climate Action (BMWK)*.

Rahmani, K., & Thomson, V. (2009). New interface management tools and strategies for complex products. *The 6th International Product Lifecycle Management Conference (PLM09)*, 10.

Rahmani, K., & Thomson, V. (2011). Managing subsystem interfaces of complex products. *International Journal of Product Lifecycle Management*, *5*(1).

Reich, J., Macke, N., & Heinecke, F. (2022). Standardisierung des Interfacekonzeptes. *Bitkom AK Industrie 4.0 Interoperabilität*.

Reich, J., Zentarra, L., & Langer, J. (2021). Industrie 4.0 und das Konzept der Verwaltungsschale–Eine kritische Auseinandersetzung. *HMD Praxis der Wirtschaftsinformatik*, *58*(3), 661–675.

Reichart, G., & Asmus, R. (2021). Progress on the AUTOSAR Adaptive Platform for Intelligent Vehicles. In T. Bertram (Ed.), *Automatisiertes fahren 2020* (pp. 67–75). Springer Fachmedien Wiesbaden.

Sabetzadeh, M., Nejati, S., Briand, L., & Mills, A.-H. E. (2011). Using sysml for modeling of safety-critical software-hardware interfaces: Guidelines and industry experience. *2011 IEEE 13th International Symposium on High-Assurance Systems Engineering*, 193–201.

Sage, A. P., & Lynch, C. L. (1998). Systems integration and architecting: An overview of principles, practices, and perspectives. *Systems Engineering: The Journal of The International Council on Systems Engineering*, *1*(3), 176–227.

Schmittner, C., Dobaj, J., Macher, G., & Brenner, E. (2020). A preliminary view on automotive cyber security management systems. *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 1634–1639.

SENSORIS Innovation Platform. (2017). SENSORIS Interface Architecture Version 1.4.0. *SENSORIS Innovation Platform hosted by ERTICO - ITS Europe*.

Shadab, N., & Salado, A. (2020). Towards an interface description template for reusing ai-enabled systems. *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2893–2900.

Shames, P. M., Sarrel, M. A., & Friedenthal, S. (2016). Modeling systems-of-systems interfaces with SysML. *14th International Conference on Space Operations*, 2500.

Sheard, S., Creel, R., Cadigan, J., Marvin, J., Chim, L., & Pafford, M. E. (2018). INCOSE Working Group Addresses System and Software Interfaces. *INSIGHT*, *21*(3), 62–71.

Spalazzese, R., Pelliccione, P., & Eklund, U. (2017). INTERO: an interoperability model for large systems. *IEEE Software*, *37*(3), 38–45.

Standardization Council I4.0. (2022). Industrie 4.0 component and the concept of the Asset Administration Shell. *IEC/TC65 WG 24*. https://www.sci40.com/english/asset-administration-shell/

Stevens, W. P., Myers, G. J., & Constantine, L. L. (1974). Structured design. *IBM Systems Journal*, *13*(2), 115–139.

Szyperski, C., Gruntz, D., & Murer, S. (2002). *Component software: Beyond object-oriented programming*. Pearson Education.

Tischer, M. (2018). AUTOSAR Adpative - Das Rechenzentrum im Fahrzeug. *Elektronik automotive Sonderausgabe Bordnetz 2018*.

Walden, D. D., Roedler, G. J., Forsberg, K., Hamelin, R. D., & Shortell, T. M. (Eds.). (2015). *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities* (4th ed.). Wiley.

Wang, W., Tolk, A., & Wang, W. (2009). The levels of conceptual interoperability model: applying systems engineering principles to M&S. *arXiv preprint arXiv:0908.0191*.

Wäschle, M., Behrendt, M., Xing, K., Shi, H., & Albers, A. (2021). Contract-based methods and activities in the validation of interfaces for System of Systems. *2021 16th International Conference of System of Systems Engineering (SoSE)*, 102–107.

Weilkiens, T. (2019). Nextgensysml part 7 – interface requirements. *Next Generation Systems Modeling Language*. https://mbse4u.com/2019/04/19/nextgensysml-part-7-interface-requirements/

Wheatcraft, L. S. (2010). Everything you wanted to know about interfaces, but were afraid to ask. *INCOSE International Symposium*, *20*(1), 1132–1149.

Woodcock, J., Cavalcanti, A., Fitzgerald, J., Larsen, P., Miyazawa, A., & Perry, S. (2012). Features of cml: A formal modelling language for systems of systems. *2012 7th International conference on system of systems engineering (SoSE)*, 1–6.

Zhou, B., Dvoryanchikova, A., Lobov, A., & Lastra, J. L. M. (2011). Modeling system of systems: A generic method based on system characteristics and interface. *2011 9th IEEE International Conference on Industrial Informatics*, 361–368.

ZVEI. (2016). Beispiele zur Verwaltungsschale der Industrie 4.0-Komponente –Basisteil. *Zentralverband Elektrotechnikund Elektronikindustrie e.V., (ZVEI)*.

A. Appendices

A.1. Complete Overview of Relevant Approaches from Literature Review

TITLE	AUTHOR(S)	PUBLISHED IN	CATEGORY	COMMENT	RELEVANCE
A Generic Model for the Specification of Software Interface Requirements and Measurement of their Functional Size	Khalid T. Al-Sarayreh, Alain Abran	Eighth ACIS International Conference on Software Engineering Research, Management and Applications, 2010	SE	Interface-related descriptions for specification of software interface requirements	○
A Survey on the Interplay between Software Engineering and Systems Engineering during SoS Architecting	Hector Cadavid, V. Andrikopoulos, P. Avgeriou, J. Klein	Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2020	SoSE	Investigation of interplay between system-level and software-level	○
A Taxonomy of Modeling Approaches for Systems-of-Systems Dynamic Architectures: Overview and Prospects	Ahmad Mohsin, Naeem Khalid Janjua, Syed M.S. Islam, Valdemar Vicente Graciano Neto	14th Annual Conference System of Systems Engineering (SoSE), 2019	SoSE	Overview of Architecture Definition Languages (ADLs) in SoS	◐
Achieving System-of-Systems Interoperability Levels Using Linked Data and Ontologies	Jakob Axelsson	INCOSE International Symposium 30(1):651-665, 2020	SoSE	Investigation on combination of LCIM and semantic web	◐
Ai concepts for system of systems dynamic interoperability	Jacob Nilsson, Saleha Javed, Kim Albertsson, Jerker Delsing, Marcus Liwicki, Fredrik Sandin	DiVA Digitala Vetenskapliga Arkivet, 2021	SoSE	Survey of AI/ML approaches for interoperability solutions in SoS	○
An Interface Specification for Principle Solutions Supporting the Cross-Domain Design of Mechatronic Systems	Stefan Möhringer,Jürgen Gausemeier.	Proceedings of DESIGN 2002, the 7th International Design Conference, Dubrovnik 2002	SE	Cross-domain interface specification in semi-formal modeling	●
Architecting systems of systems: A tertiary study	Héctor Fabio Cadavid, Vasilios Andrikopoulos, Paris Avgeriou	Information and Software TechnologyVolume 118 Issue C, Feb 2020	SoSE	Presents current research status of SoS	○
Behavioral interface specification languages	John Hatcliff, Gary T. Leavens, K. Rustan M. Leino, Peter Müller, Matthew Parkinson	ACM Computing SurveysVolume 44 Issue 3 Article No.:16 pp. 1–58, 2012	SE	Survey on behavioural interface specification languages	◐
Challenges for SoS Architecture Description	Thais Batista	SESoS '13: Proceedings of the First International Workshop on Software Engineering for Systems-of-Systems, 2013	SoSE	Summary of challenges of SoS wrt. architecture description	○
Contract-based interface specification language for functional and non-functional properties	Richard J. Payne, John S. Fitzgerald	School of Computing Science Technical Report Series Newcastle University, 2011	SoSE	SysML extension for contract-based interfaces	●

Continued on next page

TITLE	AUTHOR(S)	PUBLISHED IN	CATEGORY	COMMENT	RELEVANCE
Contract-based methods and activities in the validation of interfaces for System of Systems	Moritz Wäschle, Matthias Behrendt, Kangning Xing, Hanwen Shi, Albert Albers	16th International Conference of System of Systems Engineering (SoSE), 2021	SoSE	Contract-based validation of interfaces	◐
Data-Based System Engineering: ICDs management with SysML	Thierry Le Sergent, Alain Le Guennec	Embedded Real Time Software and Systems (ERTS2014),France, 2014	SE	SCADE tool and management of ICDs in SysML	◐
Development of an Interface Analysis Template for System Design Analysis	Amad Uddin, Felician Campean, Mohammed Khurshid Khan	Proceedings of the 20th International Conference on Engineering Design (ICED 15) Vol 4: Design for X, Design to X, Milan, Italy, 2015	SoSE	Presentation of an interface analysis template	◐
Documentation-as-Code for Interface Control Document Management in Systems of Systems: A Technical Action Research Study	Héctor Cadavid, Vasilios Andrikopoulos, Paris Avgeriou	European Conference on Software Architecture ECSA 2022: Software Architecture pp. 19–37	SoSE	Benefits of replacing document-centered interface management with documentation-as-code	●
Documenting software architecture: Documenting interfaces	Felix Bachmann, Len Bass, Paul Clements, David Garlan, James Ivers	Carnegie Mellon University Pittsburg, Software Engineering Institute, 2002	SE	Approaches to interface documentation	●
ECSS-E-ST-10-24C – Interface management	ECSS European Cooperation for Space Standardization	Active Standards of ECSS European Cooperation for Space Standardization, June 2015	SE	Standards for interface management in space domain	◐
Evaluation of architectural frameworks supporting contract-based specification	Richard Payne, John Fitzgerald	School of Computing Science Technical Report Series, 2010	SoSE	Surveys approaches to the contract-based specification of SoS-constituent systems	●
Everything you wanted to know about interfaces, but were afraid to ask	Louis S. Wheatcraft	INCOSE International Symposium 20(1):1132-1149, 2010	SE	Guidance on interface requirements	◐
Interface Management	John Blyler	IEEE Instrumentation and Measurement Magazine 7(1):32 - 37, 2014	SE	General ideas on interface management in SE	○
Interface Management- the Neglected Orphan of Systems Engineering	Paul Davies	INCOSE Internatinal Symposium Volume 30, Issue1, pp. 747-756, 2020	SE	Characteristics of interface management in SE	○
Interface specification for system-of-systems architectures	Richard Payne, Jeremy Bryans, John Fitzgerald, Steve Riddle	7th International Conference on System of Systems Engineering (SoSE), 2012	SoSE	Research challenges arising from combination of SysML with formal notations	●
Interface specification: A balancing act	Hans Jonkers	International Symposium on Component-Based Software Engineering CBSE, pp 5–6, 2004	SE	ISpec as formal approach to interface specification	◐
Model-based Interface Specification for Systems Integration in Systems of Systems Engineering	Daniele Gianni, Andrea D'Ambrogio, Pierluigi DeSimone, Marco Lisi, Michele Luglio	INCOSE International Symposium Volume 22, 2012, pp. 2040-2052	SoSE	Interface Communication Modeling Language (ICML), industry: Global Navigation Satellite Systems	●
Modeling systems-of-systems interfaces with SysML	Peter Shames, Marc A. Sarrel, Sanford Friedenthal	14th International Conference on Space Operations, 2016	SoSE	SysML modeling of SoS and their interface	●

Continued on next page

TITLE	AUTHOR(S)	PUBLISHED IN	CATEGORY	COMMENT	RELEVANCE
Modelling system of systems interface contract behaviour	Oldrich Faldik, Richard Payne, John Fitzgerald, Barbora Buhnova	Formal Engineering approaches to Software Components and Architectures, EPTCS 245, pp. 1–15, 2017	SoSE	COMPASS Modeling Language (CML)	●
NASA Reference Publication: Training Manual for Elements of Interface Definition and Control	Vincent R Lalli, Robert E. Kastner, Henry N. Hartt	NASA Reference Publication 1370, 1997	SE	NASA training manual for interface definitions	●
New interface management tools and strategies for complex products	Keyvan Rahmani, Vincent James Thomson	The 6th International Product Lifecycle Management Conference (PLM09). 2009	SE	Interface management model for multidisciplinary products	●
Semi-formal and formal interface specification for system of systems architecture	Jeremy Bryans, Richard Payne, Jon Holt, Simon Perry	IEEE International Systems Conference (SysCon), 2013	SoSE	Interface design pattern for interface specification	●
SysML contracts for systems of systems	Jeremy Bryans, John Fitzgerald, Richard Payne, Alvaro Miyazawa, Klaus Kristensen	9th International Conference on System of Systems Engineering (SOSE), 2014	SoSE	Contract Pattern for contractual specification of interfaces	●
System of Systems Design Verification: Problematic, Trends and Opportunities	Mustapha Bilal, Nicolas Daclin, Chapurlat Vincent	Enterprise Interoperability VI: Interoperability for Agility, Resilience and Plasticity of Collaborations. Springer International Publishing, 2014	SoSE	SoS design model verification	○
Systems and Software Interface Survey	Sally Muscarella, Macaulay Osaisai, Sarah Sheard	INCOSE International Symposium 30(1):1294-1312, 2020	SE	Survey on best practices and challenges on interface between systems and software	◐
Systems engineering interfaces: A model based approach	Elyse Fosse, Christopher L. Delp	IEEE Aerospace Conference, 2013	SE	Model-based engineering approach using SysML for interfaces	●
Systems interface management with MBSE: from theory to modeling to reality	Matthew Hause	INCOSE International Symposium 28(1):1012-1026, 2018	SE	Systems interface management using MBSE	◐
Towards an interface description template for reusing ai-enabled systems	Niloofar Shadab, Alejandro Salado	IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2020	SE	ICDs for ai-enabled systems	●
Towards automating Interface Control Documents elaboration and management	Hassna Louadah, Roger Champagne, Yvan Labiche	ACES-MB 2014 In conjunction with MODELS 2014	SE	Approach on automation of ICDs in avionics domain	●
Towards practical and sound interface specifications	Hans Jonkers	International Conference on Integrated Formal Methods IFM, pp. 116–135, 2000	SE	ISpec approach to interface specifications	●
Verifying interfaces and generating interface control documents for the alignment and phasing subsystem of the thirty meter telescope from a system model in SysML	Sebastian Herzig, Robert Karban, Mitchell Troy, Gary L. Brack, Frank G. Dekens	Modeling, Systems Engineering, and Project Management for Astronomy VIII, 2018	SE	Method for verification of interfaces and generation of ICDs from SysML	●

Continued on next page

TITLE	AUTHOR(S)	PUBLISHED IN	CATEGORY	COMMENT	RELEVANCE
Will the ICDs please stand up? An attempt to reason about subsystem interfaces in avionics system integration	Roger Champagne, Hassna Louadah	Architecture-Centric Virtual Integration (ACVI), 2016	SE	Research on establishing common vocabulary for ICDs in avionics	●

● = high relevance ◐ = medium relevance ○ = low relevance

A.2. Material Related to Expert Interviews

A.2.1. Research Questions

Below are the questions that were asked during the expert interviews:

Introduction - Open Questions about the Background of your Company/Projects

1. Please provide a brief characterisation of your company/projects. #application domain, #Standards used, #Software development life cycle model followed, #Systems engineering or #Systems of Systems Engineering
2. May we mention the name of your company in the study?
3. What is your organisational role and what part of the organisation do you represent?

Interface Specifications – Basics

4. How would you define an interface?
5. Which types of interfaces do you face in your project?
6. What are the use cases of interface specifications in your project? How are interface specifications contributing to your work?
7. When (in which phase in the engineering life cycle) do you use interface specifications?
8. Are you facing challenges with interface specifications? Where do you see improvement potentials?

Interface Specifications – Approach

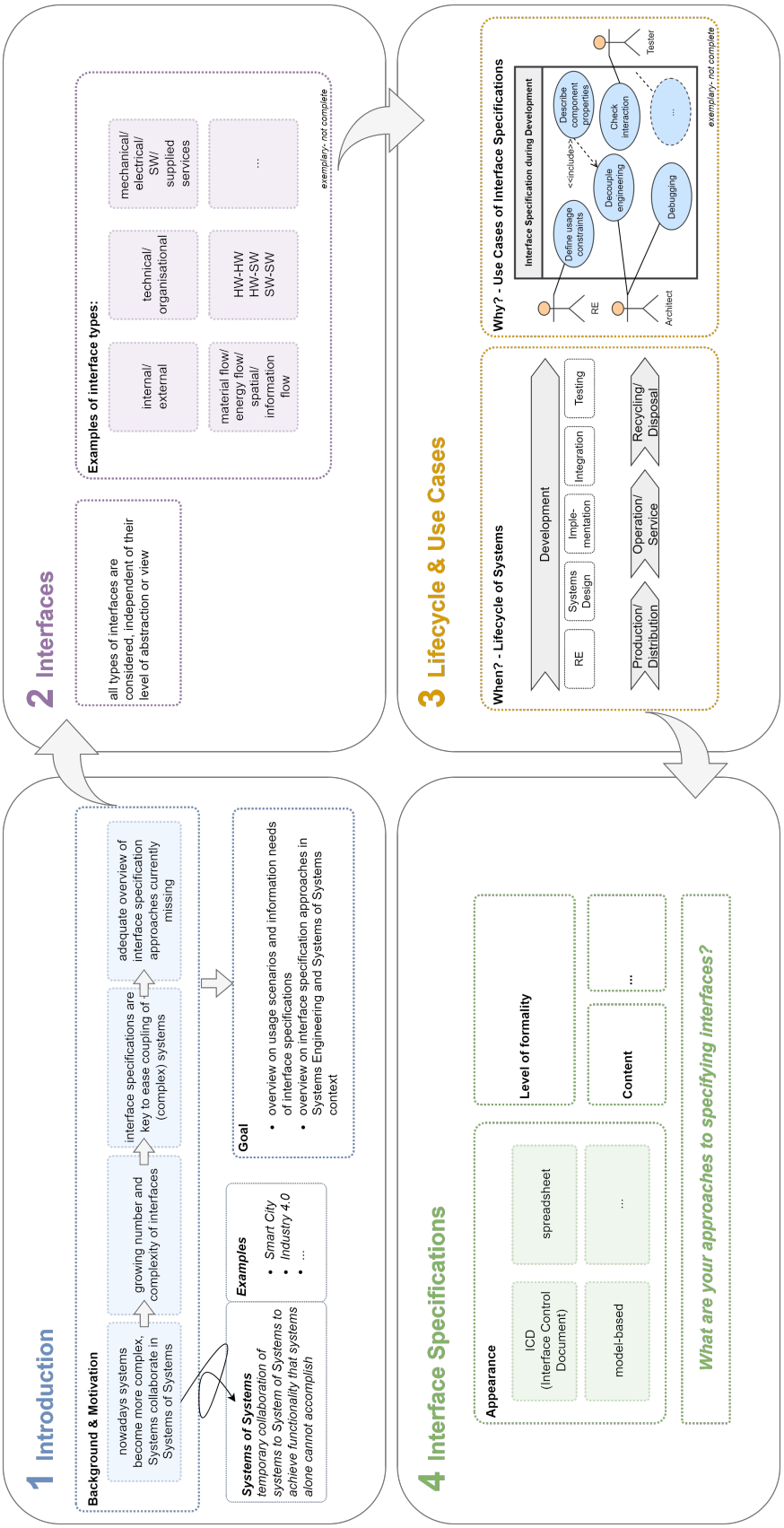
9. How are interfaces described or specified in your project and what do the interface specifications look like? (Are they e.g. purely document-centric or model-based?)
10. In case you are using MBSE-methods: Where does MBSE ease resp. complicate interface specifications?
11. Do you use contracts resp. contract-based methods in specifying interfaces?
12. How do you create and maintain interface specifications?
13. How are interface specifications created and maintained between artefacts developed in different companies? If so, what are specific challenges/ approaches to this end?
14. How do you standardise interfaces and interface specifications?
15. What happens on the change (or deletion) of an interface resp. interface specification?
16. Are any metrics used for measuring the quality of interface specifications? What does a “good” interface specification mean according to your knowledge?

Wrap Up

17. Which improvement potential do you see regarding to interface specifications within your organisation?

18. Which promising initiatives, approaches etc. do you see regarding to interface specifications?
19. Which other question would you have asked in this context?

A.2.2. Quick Start Guide from MIRO Board



A.2.3. Transcripts of Expert Interviews

A.2.3.1. Expert Interview with Interviewee ID1, Automotive Domain

The interview was declared as *Interview ID1*. It was hold on 01.02.2023 via MS Teams in German. The participants were Interviewee ID1 and Ricarda Schüssler as the interviewer. The following is the interview's transcript in a condensed format:

Interviewer (Ricarda Schüssler): Herzlich Willkommen zum Interview. Ich würde zunächst mit Hilfe des MIRO Boards kurz den Hintergrund der Arbeit erklären, um die Fragen besser einordnen zu können. [Interviewer (Ricarda Schüssler) erklärt MIRO Board: Quick Introduction]. Kommen wir zu den ersten Fragen. Hier geht es zunächst um eine Vorstellung. Frage 1: Kannst du bitte kurz dein Unternehmen bzw. dein Projekt vorstellen?

Interviewee ID1: Gerne, ich arbeite in der Automobilindustrie. In meiner Abteilung entwickeln und verkaufen wird ADAS Produkte und Airbag-Steuergeräte. Wir verfolgen einen adaptierten V-Modell Prozessansatz, der ASPICE und ISO26262 (Fusa) sowie ISO21434 (CyberSec) compliant ist.

Interviewer (Ricarda Schüssler): Dürfen wir den Namen deiner Firma in der Studie verwenden?

Interviewee ID1: Nein, bitte nicht.

Interviewer (Ricarda Schüssler): Was ist deine organisatorische Rolle im Unternehmen?

Interviewee ID1: Ich arbeite in der Rolle eines Abteilungsleiters in einer unserer acht Divisionen. Hier verantworte ich den Bereich Produkt-Conformance. Dieser besteht aus den Rollen Divisional Lead Legal compliant development, Divisional process Lead Functional Safety& Cyber Security und Lead Safety Cyber Security Management Airbag Control Units.

Interviewer (Ricarda Schüssler): Danke, das klingt nach vielen Schnittstellen! Kommen wir nun zum nächsten Fragenblock: Wie würdest du eine Schnittstelle definieren?

Interviewee ID1: Ja, richtig. Mein Arbeitsalltag ist geprägt von Schnittstellen. Für mich ist eine Schnittstelle der Berührungspunkt zwischen zwei oder mehr Objekten, die zueinander in Beziehung stehen. Hierbei kann ein Objekt alles sein. Es kann eine Komponente sein, ein Prozess, zwei zum Beispiel mechanische Bauteile, aber auch ein Mensch kann hierbei als Objekt verstanden werden.

Interviewer (Ricarda Schüssler): Spannend, dass du hier auch Menschen erwähnst. Heißt das für dich wäre auch eine Sprache oder die Kommunikation als Ganzes eine Schnittstelle?

Interviewee ID1: Ja richtig, für mich stellt dies sogar eine sehr komplexe Schnittstelle dar. In einer ISO-Norm ist beschrieben, dass es drei unterschiedliche Ausprägungen an Formalität gibt. Diese sind formal, semi-formal und informal. Bei formalen Beschreibungen

werden Zusammenhänge quasi mathematisch beschrieben. Semi-formale Sprache führt hierbei gewisse Einschränkungen bzw. Regeln an die Sprache an. Bei informaler Sprache, also der natürlichen Sprache gibt es keinerlei Regeln oder Definitionen wie etwas beschrieben wird. Daher kommt es gerade hier zu Missverständnissen und Hürden. Das Sender-Empfänger-Konzept beschreibt dies ganz gut.
Interviewer (Ricarda Schüssler): Interessant, Formalität habe ich auch als eine Eigenschaft der Schnittstellenbeschreibungen herausgearbeitet. Das deckt sich mit deiner Ausführung. Welchen Arten von Schnittstellen begegnest du in deinen Projekten?
Interviewee ID1: Das ist schwierig ganz konkret einzuteilen. Ich würde sagen Kundenschnittstellen, Prozessschnittstellen und Work-Product-Schnittstellen. In meiner früheren Position habe ich mich viel mit Hardware-Software-Schnittstellen auseinandergesetzt. Gerade bei diesen interdisziplinären Schnittstellen spielt das Thema Sprache eine große Rolle. Es ist extrem herausfordernd beispielsweise eine Geschwindigkeit in der Software, Hardware und Mechanik so zu formulieren und zu übergeben, dass am Ende der gleiche Wert raus kommt. Hier bedarf es extrem viel Abstimmung und der menschliche Faktor spielt eine riesige Rolle. Gerade Personen, die erstmal nur ihren Arbeitsbereich sehen, nur durch ihre eigene Brille schauen und wenig über den Tellerrand beispielsweise der Software oder Hardware hinaus schauen können hier hinderlich sein. Es gibt nur sehr wenige Personen, die diese unterschiedlichen Kompetenzen vereinen.
Interviewer (Ricarda Schüssler): In welchen Phasen des Lebenszyklus begegnest du Schnittstellen und Schnittstellenbeschreibungen?
Interviewee ID1: Prinzipiell erstmal im Entwicklungsumfeld und dabei vor allem auf der linken Seite des V-Modells. Teilweise bin ich hier auch in der Phase vor der eigentlichen Entwicklung, also in der Konzeptphase unterwegs. Generell zum Beispiel über Schnittstellen zur Customer Specification, Schnittstellen zwischen System-Entwicklung und Software-Entwicklung, Schnittstellen zwischen System-Entwicklung und elektronischer Hardware-Entwicklung. Zusätzlich das HSI, also das Hardware Software Interface in dem alle Berührungspunkte zwischen System, Hardware und Software spezifiziert werden.
Interviewer (Ricarda Schüssler): Okay, ich nehme aus deiner Antwort mit, dass sich Schnittstellenbeschreibungen nicht auf eine spezifische Phase beschränken. Was sind dabei die Use Cases von den Schnittstellenbeschreibungen?
Interviewee ID1: Genau. Schnittstellenbeschreibungen werden genutzt um zwischen den einzelnen Entwicklungsschritten der drei, beziehungsweise vier Disziplinen zu harmonisieren und koordinieren. Diese umfassen für mich System, Hardware, Software und eventuell Mechanik. Sie sind ein Mediator um zum Beispiel physikalische Einheiten vom z.B. Systemebene in die Software zu übersetzen. Am Beispiel der Geschwindigkeit wäre das auf Systemebene m/s in digitale Einheiten, also Anzahl der gezählten Zähne vom

Raddrehzahlsensor auf Software-Ebene.
Interviewer (Ricarda Schüssler): Gibt es dabei Herausforderungen mit Schnittstellenbeschreibungen? Siehst du hier Verbesserungspotenzial?
Interviewee ID1: Am Ende des Tages werden diese Beschreibungen ja immer von Personen erstellt und hier liegt ein Knackpunkt bzw. eine Herausforderung. Die Autoren von Schnittstellenbeschreibungen müssen unterschiedliche Skills haben, die sehr selten in einer Person kombiniert werden: tiefes Verständnis sowohl des Systems, als auch der Software und Hardware. Wichtige soft skills wie zum Beispiel die Fähigkeit offen zu kommunizieren und Kollegen zu interviewen, ein Mediator zwischen unterschiedlichen Stakeholders zu sein und gleichzeitig Spezifikationen zu einem stabilen und genehmigten Ende zu bringen.
Interviewer (Ricarda Schüssler): Interessant, das heißt du sieht Herausforderungen von Schnittstellenbeschreibungen in erster Linie in der Erstellung von diesen. Kommen wir nun zum Ansatz für Schnittstelleneschreibungen an sich. Wie sehen solche Beschreibungen bei dir aus? Sind sie z.B. modellbasiert oder dokumentenbasiert?
Interviewee ID1: Das kommt ganz auf den Anwendungsfall an. Wir dokumentieren in unserer Firma sowohl in natürlicher Sprache, als auch in semi-formalen Designsprachen wie SysML. Der Gebrauch der entsprechenden Sprache hängt ganz vom Anwendungsfall ab. Manchmal ist es einfacher etwas auszuzeichnen anstatt es mit mehr als 1000 Wörtern zu beschreiben. In anderen Anwendungsfällen würde man eventuell eher natürliche Sprache, also Englisch verwenden.
Interviewer (Ricarda Schüssler): Okay! Das heißt ihr nutzt modell-basierte Ansätze wenn der Anwendungsfall es verlangt. Gibt es ein Schema wann welcher Ansatz verwendet wird?
Interviewee ID1: Richtig, dies hat wie gerade gesagt einige Vorteile je nach Anwendungsfall. Ein offizielles Schema wann welcher Ansatz verwendet wird gibt es hier nicht, dies wird individuell entschieden.
Interviewer (Ricarda Schüssler): Verwendet ihr interface contracts?
Interviewee ID1: Ehrlich gesagt kann ich mit dem Begriff wenig anfangen. Kannst du dies kurz erklären?
Interviewer (Ricarda Schüssler): [erklärt Interface contracts]
Interviewee ID1: Danke für die Erklärung, klingt nach einem interessanten und vielversprechenden Ansatz, das geht ja auch in die formale Richtung. Ich könnte mir vorstellen, dass dies im Unternehmen verwendet wird. In meinen Bereichen bzw. Schnittstellen kenne ich diesen Ansatz nicht.
Interviewer (Ricarda Schüssler): Die nächsten zwei Fragen können wir in Anbetracht der fortgeschrittenen Zeit zusammenfassen: Wie werden Schnittstellenbeschreibungen erstellt und maintained? Wie wird dies bei Schnittstellen zwischen Unternehmen gehand-

habt?
Interviewee ID1: Grundsätzlich werden diese in rekursiven Meetings mit den betroffenen stakeholders bzw. representatives erstellt. Wenn maintenance gebraucht wird, wird hier ein Vorschlag erstellt und dieser formuliert und mit den betroffenen Parteien negotiated. Als Tool verwenden wir hier Doors. Dies bietet die Möglichkeiten von Historien, Versionierungen und Baselining. Externe Firmen erhalten hier dann entsprechende Exporte aus dem Tool.
Interviewer (Ricarda Schüssler): Wie werden Schnittstellen standardisiert?
Interviewee ID1: In manchen Bereichen haben wir gängige Templates. So zum Beispiel beim DIA oder HSI. Beim Beispiel HSI zum Beispiel ist dieses Template abgeleitet von einem Vorschlag der ISO26262 in einer Tabellenform. Das DIA Template ist so auch schon lange in Nutzung und erfährt nur minimale Updates. Der DIA ist auch in Tabellenform geführt. Die Tabelle an sich ist immer gleich aufgebaut und enthält den eigentlichen Inhalt, ein Verzeichnis zu den Abkürzungen und eine Revisionshistorie. Wichtig zu erwähnen hier ist die Semantik, die im Abkürzungsverzeichnis zu tragen kommt. Wie eingangs erwähnt müssen immer alle Beteiligten wissen wovon sie sprechen und das gleiche Verständnis haben.
Interviewer (Ricarda Schüssler): Gibt es Metriken um die Qualität von Schnittstellenbeschreibungen zu messen? Was bedeutet eine „gute Schnittstellenbeschreibung“ für dich?
Interviewee ID1: Metriken zum Messen der Qualität gibt es hier keine. Der zweite Fragenteil ist herausfordernd. Für mich sollte eine gute Schnittstellenbeschreibung nur so viel Informationen wie nötig enthalten aber gleichzeitig auch alle relevanten Informationen zur Verfügung stellen. Ich stelle mir so etwas gerne bildlich vor und da ich ursprünglich aus dem Maschinenbau-Umfeld komme ist für mich ist eine Schnittstellenbeschreibung wie ein guter Kleber, der zwei Komponenten zusammenbringt und vereint.
Interviewer (Ricarda Schüssler): Siehst du Verbesserungspotenzial im Hinblick auf Schnittstellenbeschreibungen?
Interviewee ID1: Das ist auch wieder total abhängig vom Anwendungsfall beziehungsweise von der Schnittstelle, die betrachtet wird. Ich habe im Interview schon einiges zum DIA gesagt. Dieser ist für mich als Schnittstellenbeschreibung in dem speziellen Anwendungsbereich so ausgereift und bedarf keinerlei Verbesserung. Alle beteiligten Parteien wissen hier genau was sie tun müssen und wie sie diese Schnittstelle festhalten. Das Vorgehen funktioniert in der Praxis wirklich super. Eine Stelle, an der ich Verbesserungspotenzial sehe, ist das Prinzip von reuse bzw. information hiding, wie dies auf Komponentenebene gelebt wird. Hier können einzelne Komponenten gegeneinander ausgetauscht werden, ohne dass ein Außenstehender in die Komponente hineinsehen kann. Rein über die Schnittstellenbeschreibung gibt es hier eine Aussagekraft zu der entsprechenden Komponente,

die Komponente selbst ist eine black box. Die Komponenten können dann als black box wiederverwendet werden. Um nochmals zurück zur bildlichen Vorstellung zu kommen: die Komponenten können wie Legosteine aneinander gesetzt werden Dieses Prinzip würde ich mir auch bei unseren übrigen Schnittstellen, beispielsweise im Prozessumfeld wünschen. Eventuell würde dies über eine Standardisierung funktionieren.
Interviewer (Ricarda Schüssler): Spannend, kennst du hier Ambitionen in diese Richtung?
Interviewee ID1: Bislang eher weniger, ich denke aber dass dieses Thema in Zukunft von immer größerer Bedeutung sein wird.
Interviewer (Ricarda Schüssler): Ja, ganz bestimmt. Welche übrigen Fragen hast du bzw. hättest du in dem Kontext Schnittstellenbeschreibungen gestellt?
Interviewee ID1: Was ich zu dem Thema sagen möchte ist, dass es extrem vielschichtig ist und daher sehr herausfordernd hier einen gemeinsamen Nenner zu finden. Während der Vorbereitung auf dieses Interview fiel es mir relativ schwer hier allgemeingültige Aussagen zu Schnittstellen treffen zu können. Allein die Einordnung mit welcher Schnittstelle ich mich gerade beschäftige ist meiner Meinung nach essenziell, da es extrem viele Abhängigkeiten gibt.
Interviewer (Ricarda Schüssler): Ja, das ist auch eine meiner Erkenntnisse während dieser Thesis. Mit Hilfe der Arbeit möchte ich ein Schema zur Einordnung erstellen, das die unterschiedlichen Dimensionen von Schnittstellenbeschreibungen darstellt. Auf dieser Basis kann man dann eventuell Cluster finden oder näher auf einzelne Probleme eingehen.
Interviewee ID1: Das macht durchaus Sinn.
Interviewer (Ricarda Schüssler): Vielen Dank für deinen Input, deine Zeit und das Interview!

A.2.3.2. Expert Interview with Interviewee ID2, Automotive Domain

The interview was declared as *Interview ID2*. It was hold on 03.02.2023 via MS Teams in German. The participants were Interviewee ID2 and Ricarda Schüssler as the interviewer. The following is the interview's transcript in a condensed format:
Interviewer (Ricarda Schüssler): Herzlich willkommen zum Interview! Bevor wir auf die Fragen eingehen würde ich gerne kurz auf den Hintergrund und die Ziele der Arbeit eingehen [Interviewer (Ricarda Schüssler erklärt QuickStartGuide auf MIRO Board]. Kommen wir nun zum Fragebogen. Kannst du dich bitte kurz vorstellen: In welcher Domäne und nach welchen Prozessen bzw. Standards arbeitest du?
Interviewee ID2: Ich bin Applikationsingenieur in der Automobilindustrie und bin dort verantwortlich für die funktionale Sicherheit des Motorsteuergeräts. In der Vergangenheit war ich für Projekte mit Verbrennungsmotoren verantwortlich, mittlerweile ausschließlich

für EV-Projekte. Zusätzlich bin ich weltweiter Ansprechpartner und Center of Competence in diesem Bereich. Wir verfolgen einen V-Modell Ansatz und im Rahmen der funktionalen Sicherheit stützen wir uns auf die ISO26262.
Interviewer (Ricarda Schüssler): Dürfen wir den Namen deiner Firma in den Ergebnissen erwähnen?
Interviewee ID2: Mir wäre lieber, wenn er nicht erwähnt wird.
Interviewer (Ricarda Schüssler): Was ist deine Rolle im Unternehmen?
Interviewee ID2: Innerhalb der Kundenprojekte bin ich für unser zentrales Steuergerät für einen speziellen Kunden zuständig und appliziere hier die Überwachung der Funktionsebene. Zusätzlich trage ich dafür Sorge, dass das Fahrzeug sich immer in einem sicheren Zustand befindet, auch während einer Fehlfunktion. Ich baue hierbei auf die Software auf.
Interviewer (Ricarda Schüssler): Danke für die Vorstellung! Kommen wir nun zum nächsten Fragenteil: Wie würdest du ein interface bzw. eine Schnittstelle definieren?
Interviewee ID2: Für mich ist eine Schnittstelle im Endeffekt eine Barriere, die überwunden werden muss. Hierbei sollen Informationen von einem System in ein anderes transportiert werden. Ein System kann für mich auch eine Person sein.
Interviewer (Ricarda Schüssler): Interessant, dass du den Begriff „Barriere“ verwendest. Dieser ist im Sprachgebrauch ja erstmal negativ konnotiert.
Interviewee ID2: Ja, ich sehe dies tatsächlich als Barriere und verwende den Begriff bewusst. Man muss sich im Bereich Schnittstellen schon vielfältige Gedanken machen.
Interviewer (Ricarda Schüssler): Welchen Arten von Schnittstellen begegnest du in deinem Projekt?
Interviewee ID2: Prinzipiell begegnen mir sehr verschieden Schnittstellen. Wie eben erwähnt habe ich organisatorische bzw. fachliche Schnittstellen zur Softwareentwicklung, gerade im Bereich Umsetzung funktionaler requirements. Außerdem habe ich prozessorale Schnittstellen zur ISO26262, die erfüllt werden müssen. Hier werden uns beispielsweise Methoden und (rechtliche) Regeln vorgegeben. Zusätzlich habe ich Schnittstellen zum Projektteam und zum Kunden an sich, gerade im Bereich Abstimmung der Funktionen und Fehlersuchen. Außerdem begegne ich Software-Schnittstellen zwischen Überwachungsebene und Funktionsebene. „Meine“ Überwachungsebene greift mit sogenannten handshakes in die Funktionsebene ein, um diese zu überprüfen und im Fehlerfall die Sicherheit zu gewährleisten. Prinzipiell ist dies also eine Schnittstelle zwischen einzelnen Funktionen auf dem Steuergerät. Wir lehnen uns hier mit der Überwachungsebene an die Funktionsebene und sind somit direkt von ihr abhängig.
Interviewer (Ricarda Schüssler): Was ist der handshake hierbei?
Interviewee ID2: Jede Funktion besitzt Eingangs- und Ausgangssignale. Ganzheitlich betrachtet hat das Motorsteuergerät Schnittstellen zum restlichen Fahrzeug, z.B. über

CAN. Innerhalb des Steuergeräts sind die einzelnen Funktionen miteinander verknüpft und werden von der Funktions- und Überwachungsebene verarbeitet. Zwischen der Funktions- und Überwachungsebene kommen die sogenannten handshakes ins Spiel. Dies sind Ausgangssignale aus der Überwachungs- in die Funktionsebene.
Interviewer (Ricarda Schüssler): Sind dies feste Werte, die übergeben werden oder kommen hier auch Bedingungen mit ins Spiel. Ich Frage, weil dies in Richtung interface contracts gehen würde.
Interviewee ID2: Nein, es werden hier skalare oder binäre Werte übergeben. Bedingungen werden an der Schnittstelle weniger übergeben. Ein reiner Datenfluss also.
Interviewer (Ricarda Schüssler): In welcher Phase des Lebenszyklus begegnen dir Schnittstellenbeschreibungen?
Interviewee ID2: Generell befinde ich mich in der Entwicklung und habe außerhalb des Entwicklungszyklus kaum Berührungspunkte zu Schnittstellen. Innerhalb der Entwicklung habe ich unterschiedliche Berührungspunkte im V-Modell, hier insbesondere im unteren und rechten Ast.
Interviewer (Ricarda Schüssler): Was sind dabei dann deine Use Cases von Schnittstellenbeschreibungen bzw. wie helfen dir Schnittstellenbeschreibungen?
Interviewee ID2: Use Cases sind im Endeffekt zum einen ein Informationsaustausch zwischen SW, Projekt und Kunde. Außerdem die Übergabe von Daten in unserer modellbasierten Bedatungsstrategie.
Interviewer (Ricarda Schüssler): Nutz ihr diese eventuell auch zur Fehlersuche?
Interviewee ID2: Ja, auf jeden Fall. Gerade im Fahrzeugtest tritt dieser Anwendungsfall häufig auf.
Interviewer (Ricarda Schüssler): Gibt es Herausforderungen mit Schnittstellenbeschreibungen in deinem Umfeld?
Interviewee ID2: Bei unserem modellbasierten Bedatungsmodellen können die Parameter der Schnittstellen bearbeitet werden. Wir geben quasi den Typ (Skalar, Bit, Maps, Curves etc.) und die Einheit vor, die dann standardisiert wird. Im Endeffekt müssen wir die Einheit des Inputs vorgeben. Das Interface standardisiert diese Schnittstelle auf Standardeinheiten. Zusätzlich geben wir Namen der Schnittstellenein- und ausgänge vor.
Interviewer (Ricarda Schüssler): Spannend! Das bedeutet beispielsweise, dass wenn ich eine Fahrzeuggeschwindigkeit in km/h als Eingang habe und dies auf das Interface gebe, dann rechnet das Interface den Wert in m/s um?
Interviewee ID2: Richtig, ich gebe die Einheit des Inputs vor und das interface rechnet dies dann um. Dies macht das ganze recht fehleranfällig. Füttere ich das Interface mit falschen Informationen rechnet mein gesamtes Model mit den falschen Einheiten. Im Tool wird dies angezeigt und innerhalb unserer Überwachungsmodelle wird dann nur eine

Einheit verwendet. Wir verwenden hier eigens angefertigte Interface Blöcke. Prinzipiell ist dies eine Verbesserung immer standardisierte Einheiten innerhalb unserer Modelle zu verwenden allerdings muss man hier höllisch aufpassen. Gerade auch da Überwachungs- und Funktionsebene teilweise unterschiedliche Einheiten verwenden.
Interviewer (Ricarda Schüssler): Das klingt nach einem Stolperstein. Es gibt ja auch genügend prominente Beispiele in denen Projekte an Einheitenumrechnungen gescheitert sind.
Interviewee ID2: Ja, total. Über Regeln könnte man dies eventuell abfangen. Dies sehe ich als Verbesserungspotenzial. Oder, dass die Interfaces automatisiert erkennen können welche Einheit das Input-Signal innehat. Ein manuelles 4-Augen-Review hilft aber kann nicht die Lösung an dieser Stelle sein. Ein weiteres generelles Verbesserungspotenzial im Hinblick auf Schnittstellen sehe ich in der Abstimmung zwischen den einzelnen Disziplinen bzw. Gewerken im V-Modell.
Interviewer (Ricarda Schüssler): Wie sehen Schnittstellenbeschreibungen generell bei euch aus?
Interviewee ID2: Je nach Anwendungsfall verwenden wir wie angedeutet unterschiedliche Formen. Teilweise verwende wir Dokumente, teilweise eben einen modellbasierten Ansatz zur Bedatung. Wie eben schon angedeutet werden in den Modellen einzelne Module aneinander verknüpft und Skalare oder Werte über sogenannte Interface Blöcke übergeben. Es wird im Projektumfeld außerdem natürlich auch vieles über gesprochene Sprache über Schnittstellen hinweg übergeben.
Interviewer (Ricarda Schüssler): Wo erschwert bzw. verbessert der modellbasierte Ansatz euch bei Schnittstellenbeschreibungen?
Interviewee ID2: Natürlich in der Wiederverwendbarkeit und der einfachen Anpassung von Modellen bei Änderungen der Projektanforderungen. Außerdem können einzelne Modelle einfach gegeneinander wie in einem Baukastenprinzip ausgetauscht werden. Natürlich auch wie eben besprochen die Umrechnung der Einheiten klappt in einem Modell vergleichsweise simpel.
Interviewer (Ricarda Schüssler): Klar! Wir haben eben schon etwas darüber gesprochen: verwendet ihr interface contracts? [Interviewer (Ricarda Schüssler): erklärt interface contracts mithilfe von MIRO Board]
Interviewee ID2: In meinen Modellen benutze ich diesen Ansatz nicht aktiv. Ich weiß zum Beispiel, dass meine Interface Blöcke bei „falschem“ Eingang eines Typs, also wenn z.B. das Interface als Wert definiert ist und dann aber ein Kennfeld als Eingang verwendet wird, dann entspricht dies nicht den Erwartungen des Interfaces und es wird ein Fehler ausgegeben. Das Gesamtmodell kann ich dann nicht kompilieren. Ich möchte zusätzlich nicht ausschließen, dass Interface Contracts vermehrt bzw. spezieller in anderen Teilen der

Entwicklung genutzt werden.
Interviewer (Ricarda Schüssler): Wie werden Schnittstellenbeschreibungen erstellt und gewartet?
Interviewee ID2: Modellbasiert läuft dies im Tool ab, hier gibt es eine Versionierung. Ich selbst betreibe hier keine aktive maintenance.
Interviewer (Ricarda Schüssler): Wie standardisiert ihr Schnittstellen und deren Beschreibungen?
Interviewee ID2: Prinzipiell haben wir hier ja schon drüber gesprochen. Beispielsweise über die Einheiten standardisieren wir hier viel. Wichtig für mich ist hier außerdem, dass Schnittstellen einfach gehalten werden und nicht zu viel Rechenzeit benötigen. Außerdem sollten sie optisch so dargestellt sein, dass sie leicht zu bedienen sind. Dies haben wir in eigens angefertigten Schnittstellenelementen realisiert.
Interviewer (Ricarda Schüssler): Was passiert mit einer Schnittstelle wenn diese gelöscht oder bearbeitet wird?
Interviewee ID2: Prinzipiell läuft dies auch alles über das modellbasierte Tool.
Interviewer (Ricarda Schüssler): Gibt es Metriken zur Bestimmung der Qualität von Schnittstellen? Was bedeutet für dich eine gute Beschreibung einer Schnittstelle?
Interviewee ID2: Metriken dazu kenne ich nicht. Für mich sollte eine Schnittstelle eineindeutig beschrieben sein, sodass kein Zweifel daran besteht wie die Eingangs- und Ausgangssignale zu interpretieren sind und es hier zu keinen Fehlern kommen kann.
Interviewer (Ricarda Schüssler): Danke! Dann kommen wir zum letzten Fragenteil, zum Wrap Up: Welches Verbesserungspotenzial siehst du im Hinblick auf Schnittstellenbeschreibugnen?
Interviewee ID2: Ich denke da habe ich nichts zu dem hinzuzufügen, was wir bereits besprochen haben.
Interviewer (Ricarda Schüssler): Bist du in dem Zusammenhang auf weitere interessante Ansätze gestoßen?
Interviewee ID2: Adhoc fällt mir hier nichts zu ein.
Interviewer (Ricarda Schüssler): Okay! Hättest du in dem Zusammenhang weitere Fragen gestellt?
Interviewee ID2: Ich fand die Zusammenstellung der Fragen schon sehr umfangreich und umfassend. Das Thema ist extrem vielschichtig, daher finde ich es schwierig dies auf einen kleinen gemeinsamen Nenner zu bringen.
Interviewer (Ricarda Schüssler): Ja, total! Danke für deinen Input, deine Zeit und das Interview!

A.2.3.3. Expert Interview with Interviewee ID3, Agriculture Domain

The interview was declared as *Interview ID3*. It was hold on 09.02.2023 via MS Teams in German. The participants were Interviewee ID3 and Martin Becker, Florian Balduf and Ricarda Schüssler as the interviewers. The following is the interview's transcript:

Interviewer (Ricarda Schüssler): Herzlich Willkommen zum Interview. Ich möchte als Einführung kurz über den Hintergrund und die Ziele der Arbeit sprechen, um einen Rahmen für die Inhalte des Interviews abzustecken [Ricarda Schüssler erklärt Quick Start Guide in MIRO Board]. Dann würde ich gerne mit dem Fragebogen beginnen. Der erste Block wäre die Frage zu deinem Hintergrund. In welcher Applikationsdomäne befindest du dich, welche Standards werden da vielleicht verwendet? Nach welchem Prozess Modell wird gearbeitet?

Interviewee ID3: Die Applikationsdomäne in der ich jetzt bin ist Agriculture, also landwirtschaftliche Maschinen. Meine Firma ist Claas, die sich mit dem Landmaschinenbau beschäftigen. Die Standards, die für uns gelten, das sind verschiedenste: 15288, 12207, Automotive Spice (15504). Außerden haben wir etwas wie funktionale Sicherheit für agriculture Maschinen, ähnlich wie die ISI26262 für Automotive. Wir haben die Cyber Security /Cyber Security resilience act. Das sind die Dinge, die uns im Moment treiben. Das sind auch die Normen, die dazu führen, dass wir immer mehr Aufwand treiben müssen, um diese Formalien zu erfüllen, um die Maschinen zulassen zu dürfen. Das heißt, je mehr Normen wir haben- wie jetzt zum Beispiel im Bereich Cyber Security steigt unser Aufwand an Dokumentation und damit auch der Druck von einer dokumenten-zentrierten Arbeitsweise auf eine immer mehr modellbasierte Arbeitsweise umzustellen um die Ergebnisse, die wir da haben, konsistent zu halten. Insbesondere mit der Cyber Security kommt der Aspekt hinzu, dass wir sozusagen keinen Safe State mehr haben wie wir es bei funktionaler Sicherheit haben. Das heißt, wir müssen im Prinzip uns darauf einstellen, dass uns morgen jemand sagt, wir haben mit eurer Landmaschine ein Problem gefunden und möchten, dass ihr das abstellt aus Sicht der Regulierungsbehörde. Jetzt sind wir nicht Automotive, das heißt, wir haben vielleicht eine andere Kritikalität und auch eine andere Häufigkeit. Nicht so viele Fahrzeuge und so weiter, aber wir müssen uns damit auseinandersetzen, dass wir im Prinzip jederzeit nach SOP irgendwann so eine Rettungsaktion machen müssen. Softwareeingriffe also Software nachschießen müssen, die auch nach 5 Jahren 10 Jahren, nachdem die Produkte ihren SOP hatten. Das macht uns Sorgen, das macht Ärger und das ist nennenswert blöd. Das ist die Realität, auf die wir jetzt zusteuern.

Interviewer (Ricarda Schüssler): Ja, das schießt den Aufwand ziemlich in die Höhe wahrscheinlich.

Interviewee ID3: Ja, der Aufwand ist die eine Geschichte die andere Frage ist nach Reaktionsgeschwindigkeit. Wenn ich jetzt vom Gesetzgeber eine Vorgabe bekomme, der

mir sagt, du musst die Maschinen innerhalb von dieser oder jener Zeit umrüsten. Dann tickt irgendwann die Uhr und dann ist die Frage wie lange brauche ich dafür: Woche? Monat? Jahr? Der Aufwand in Manpower oder Geld ist hier gar nicht unbedingt das Ding. Die Frage ist tatsächlich: Kriegt man es in einer endlichen Zeit überhaupt hin.
Interviewer (Ricarda Schüssler): Total spannend! Dürfen wir den Namen Claas in der Studie verwenden oder wäre es dir lieber, das außen vor zu lassen?
Interviewee ID3: Das könnt ihr ruhig machen, das ist in Ordnung.
Interviewer (Ricarda Schüssler): Okay, super, danke. Dann wäre die nächste Frage: Was ist deine Rolle im Unternehmen oder für was bist du verantwortlich in dem Zusammenhang?
Interviewee ID3: Ich bin als Systems Engineering Coach im Bereich der Prozesse und Methoden/ Tools dafür verantwortlich, dass meine Kollegen das Engineering so anwenden, dass es bestimmten Regeln entspricht, die uns vor Schaden schützen durch Fehlfunktionen und durch Belange durch den Gesetzgeber. Wir müssen ein Mindestmaß an Konformität erreichen.
Interviewer (Ricarda Schüssler): Danke! Das hilft in der Einordnung der Fragen. Dann würde ich jetzt zum technischen Teil kommen. Die erste Frage wäre: Was verstehst du unter einer Schnittstelle oder wie würdest du eine Schnittstelle definieren?
Interviewee ID3: Wie würde ich das definieren? Für mich ist das der Weg über den zwei Systemelemente miteinander in Austausch kommen, abstrakt gesprochen.
Interviewer (Ricarda Schüssler): Wäre „Weg“ hier rein räumlich gesehen?
Interviewee ID3: Nein, nein, das wäre jetzt im Prinzip das Kopplungsmedium oder aber im Prinzip die Verbindung zwischen zwei Systemelementen: wie sind die miteinander verbunden und worüber können sie sich dadurch austauschen?
Interviewer (Ricarda Schüssler): Danke! Es gibt hier ein Menge an unterschiedlichen Schnittstellendefinitionen. Welche Typen an Interfaces begegnest du in deinen Projekten oder in deinem Projektalltag?
Interviewee ID3: Schnittstellen, die für uns eine Rolle spielen, sind einerseits Netzwerkschnittstellen zwischen den Steuergeräten auf den Maschinen, also Kommunikation von Steuergerät zu Steuergerät. Dann haben wir Datenfunk, das ist eine wichtige Schnittstelle von der Maschine ins Backend über UMTS oder andere Kanäle. GPS ist ein ganz wichtiger Punkt, aber auch etwas wie LIDAR- Schnittstellen, optische Schnittstellen, Radarsysteme, die die Umgebung überwachen. Alles, was mit Sensorik zu tun hat. Kameratechnik ist ein Thema. Natürlich auch mechanische Schnittstellen, zum Beispiel die Tröte vom Häcksler, die das Material in den Trailer schmeißt. Da muss man zum Beispiel den den Trailer erkennen, damit die Tröte auch das Material dort rein und nicht daneben wirft. Das ist zum Beispiel auch eine Schnittstelle, die über eine Kameratechnik funktioniert, aber die

am Ende die Mechanik dynamisch erkennen muss und die Tröte nachregeln muss. Das sind so Schnittstellen mit denen wir zu tun haben.
Interviewer (Ricarda Schüssler): Das ist sehr breit gefächert!
Interviewee ID3: Ja, wir hätten noch hydraulische Schnittstellen aber damit habe ich nichts zu tun, ich bin mehr für das E-und E- System zuständig und würde mich darauf fokussieren. Hydraulik ist natürlich bei einer Landmaschine auch ein wichtiges Thema, weil diese viel Kraft übertragen kann und auch unsere Fahrantriebe damit teilweise funktionieren. Und natürlich das Mensch-Maschine Interface. Das ist ein ganz wichtiges Thema. Wir haben auf der Landmaschine große Terminals, über die der Fahrer mit der Landmaschine kommuniziert. Und da passiert, würde ich sagen, deutlich mehr als im Fahrzeug im normalen PKW, weil ja nicht nur das Fahrgeschäft, sondern auch die ganze Wertschöpfung auf dem Feld über die Terminals mit abgebildet wird: Wie ist die Maschine eingestellt? Was hat sie gerade für Ertragswerte? Was sind die Feldgrenzen, wo muss sie was machen? Das ist nochmal eine ganz andere Welt, was da mit dazukommt und was Fahrzeuge so nicht kennen. Für den Fahrer ist dies ein sehr anspruchsvolles Mensch-Maschine Interface.
Interviewer (Ricarda Schüssler): Das heißt, ich habe als Nutzer dann recht viele Parameter, die ich an der Maschine selbst einstellen kann oder wie kann man sich das vorstellen?
Interviewee ID3: Ja, ich kann sehr viele Sachen einstellen, sagen wir zum Beispiel die Fahrgeschwindigkeit wie die Implements, die wir dran haben an den Maschinen, eingestellt sind. Mit welchen Maschinenparametern gerade gefahren wird. Vieles macht die Maschine automatisch, aber es wird angezeigt. Der Operator kann jederzeit eingreifen und selbst noch Dinge beeinflussen. Und es gibt unterschiedliche Strategien, was man will: will derjenige möglichst schnell möglichst viel vom Feld runterholen, weil das Wetter schlecht ist oder hat er Zeit und will eine möglichst gute Kornqualität erhalten. Dann muss er vielleicht langsamer fahren. Solche Parameter hat man da zum Spielen. Es ist also ein anspruchsvolles Mensch-Maschine-Interface. Und auch da muss man sagen, läuft das Ganze immer auch mit einer Backend-Verbindung, die quasi permanent Datenfunk von der Maschine in die Backend-Systeme und auch in die Vernetzung der Maschine drum herum macht. Wenn man sich jetzt einen Häcksler vorstellt, der macht in ungefähr 5 Minuten einen Trailer voll. Das heißt dann muss alle 5 Minuten ein Fahrzeug bereitstehen, das das Erntegut aufnimmt und dieses muss auch das Zeug wegschaffen. Das heißt, das ist ein großes Getriebe, die alle über Funk irgendwie miteinander verbunden sind.Denn wenn der eine nicht kann, dann kann der andere anhalten. Das ist also eine Kette also so ein Zusammenspiel was da funktionieren muss und das passiert in Echtzeit natürlich.
Interviewer (Ricarda Schüssler): Ja das klingt ziemlich vielen Schnittstellen.

Interviewee ID3: Und natürlich dabei auch das Thema mit Software Freigaben, also mit Feature- Freigaben. Man kann zum Beispiel jetzt bei Claas über die Claas App bestimmte Software Features on the fly freischalten. Wenn man beispielsweise sagt ich möchte heute dieses Zeugs ernten und möchte mir dazu die Unterstützung holen, dann kauf ich mir quasi diese Funktion, vielleicht für eine Erntesaison. Das kann ich beispielsweise morgens machen und kann sie danach nutzen.
Interviewer (Ricarda Schüssler): Das heißt sind die Landmaschinen dann mit der Claas App verknüpft oder wie kann sich das vorstellen?
Interviewee ID3: Man kann sich das so vorstellen, dass es Backend-Systeme gibt, die man zum Beispiel über mobile Geräte anfragen kann. Man kann sich einloggen und kann dann zum Beispiel sagen ich möchte mir jetzt diese Optimierungssoftware kaufen für diesen Ernteeinsatz und drücke quasi auf einen Bestellknopf. Dann wird diese Software gekauft und für die Landmaschine für diesen Einsatz freigeschaltet.
Interviewer (Ricarda Schüssler): Okay, ja, das ist aus Automobilindustriesicht schwierig nachzuvollziehen.
Interviewee ID3: Ja, das wäre im Prinzip, als würde man aufs Handy gehen und sagen können "ich habe hier zwar die Standard-Variante von dem Fahrzeug, aber ich weiß die hat eine Sitzheizung verbaut. Ich bezahle jetzt nochmal 50€ und schalte dann die Sitzheizung frei". Dann wird quasi dieses Feature am Steuergerät freigeschaltet und auf einmal kann das Fahrzeug auch Sitzheizung oder das, was sonst dort verbaut ist. Aber dahinter braucht man Verschlüsselung und ein Abrechnungssystem und all das ganze Zeug. Das ist alles am Ende in diese Schnittstelle oder in dieses Ding mit eingezogen, wenn man so will.
Interviewer (Ricarda Schüssler): Ja, das stelle ich mir eben herausfordernd vor also das klingt erstmal einfach zu sagen ich kauf mir das über den App Store und lade mir dann runter aber alles, was dann noch mit dranhängt klingt herausfordernd.
Interviewee ID3: Das ist auch nicht nur ein Klick, den man da braucht. Das sind ganz viele Klicks, aber abstrahiert geht das. Spannend wird das jetzt zum Beispiel, wenn ich die Maschine verkaufen möchte oder wenn die Maschine verschrottet wird: Wird dann die Lizenz mit verschrottet oder wenn ich die Maschine verkaufe, übertrage ich die Lizenz auf den nächsten User oder ist die an den User gebunden? Das sind also viele Sachen, die man im Detail sich genau anschauen sollte. Und du hast ja auch hier möglicherweise personengebundene Daten auf der Maschine, zum Beispiel über Apple Carplay oder irgendein anderes Zeugs, Telefonnummern und sonstige Dinge auf der Maschine. Darf ich die mitverkaufen? Das ist vielleicht auch irgendwo das Thema Schnittstelle. Das macht es kompliziert für uns. Wir verdienen da am Ende nichts dran, denn das macht die Ernte nicht besser, aber ist für den Komfort wichtig.
Interviewer (Ricarda Schüssler): Gibt es eine bestimmte Phase, in der Schnittstellen-

beschreibungen verwendet werden? Kannst du das einteilen?

Interviewee ID3: Eigentlich möchten wir Schnittstellendefinitionen am Anfang der Projekte haben. Für die Mechanik ist das teilweise schwierig, weil man teilweise am Anfang eines Projektes vielleicht noch gar nicht die mechanischen Dimensionen kennt. Elektrisch gesehen wissen wir aber doch häufig einigermaßen genau, was mit wem kommuniziert. Da brauchen wir nicht die genauen Abmaße, das können wir schon anhand der Datenstrukturen wissen wer mit wem kommunizieren muss. Das heißt, da machen wir am Anfang eines Projektes zunehmend und immer mehr Kontextanalysen, Use Case Analysen und versuchen im Kontext aufzubrechen: Wer muss mit wem kommunizieren? Und um das noch mal tiefer zu legen, machen wir das auch jetzt mit dem Concept auf Operations. Also in welcher Phase befindet sich jetzt gerade das Projekt im Life cycle? Ich bin zum Beispiel bin der Entwicklungs- und der Definitionsphase und dann ist die Frage zum Beispiel: Fokussiere ich mich auf die normale Anwendung des Systems oder betrachte ich auch was macht das System während Produktion, Transport, Werkstatt oder bei so einem autonomen System: Wie wird es für seinen Einsatz vorbereitet? Wie führt es den Einsatz durch, wie wird es nachbereitet? Was bedeutet das für die verschiedenen Systemkomponenten? Was für Modes und Zustände kennen die? Wenn man das als System of Systems denkt, dann hat man da schon ne ganze Menge zu tun. Und wenn man das versäumt sich darüber Gedanken zu machen, dann dürfte es sehr schwierig werden, das Verhalten dieses Systems hinterher für die Entwickler zugänglich zu machen. Ja, also, das gehört für uns immer ganz an den Anfang des Projektes. Es hat eine gewisse Dynamik, die sollte aber eigentlich nicht zu groß sein.

Interviewer (Ricarda Schüssler): Okay, das heißt, ihr würdet am Anfang des Projekts in dem Bereich die Schnittstellenbeschreibung aufsetzen und die dann aber während der gesamten Entwicklungsphase möglichst so nutzen, wie sie besteht, außer mit kleinen Modifikationen.

Interviewee ID3: Ja, so ist die Idee. Wenn wir das nicht machen, dann haben wir große Probleme, weil wir das System nicht an einem Standort entwickeln, sondern standortübergreifend. Das heißt, wenn wir die Schnittstellen ändern, müssen wir alle betroffenen Standorte entsprechend mit informieren. Wir müssen die Spezifikationen anpassen. Das macht dann im Detail sehr viel Ärger und Arbeit und es ist fehleranfällig, weil es vielleicht doch irgendwo nicht richtig berücksichtigt wurde. Insofern am besten am Anfang und dann das stabil halten, eventuell erweitern. So ist die Idee.

Interviewer (Ricarda Schüssler): Was macht ihr dann mit den Schnittstellenbeschreibungen? Also was sind deren Use Cases? Das würde uns besonders interessieren.

Interviewee ID3: Die Schnittstellenbeschreibungen sind am Ende an vielen Stellen wichtig. Einerseits brauchen wir die zu Beginn der Entwicklung um überhaupt Verständ-

nismodelle in den Kopf zu kriegen. Das heißt ich möchte ja erst mal wissen mit wem habe ich es überhaupt zu tun? Und dann habe ich solche Use Case-Diagramme oder Kontextdiagramme, die sehr schnell sehr viel größer werden, als mir lieb ist. Das ist nur der erste Schritt. Also erstmal brauche ich das, um überhaupt ein Bild meines Systems und ein Verständnisbild meines Systems zu erzeugen. Das andere ist: Woher weißt denn jetzt der Ingenieurskollege, was er machen soll? Zum Beispiel: Wenn wir jetzt sagen, wir entwickeln die Maschine, also den mechanischen Teil von so einer Maschine und den elektrischen Part von der Maschine. Dann muss ich ja irgendwo die Idee haben wo das denn verbaut sein soll. Ist das Steuergerät, wie schon seit den letzten hundert Jahren an der gleichen Stelle oder vielleicht woanders? Also ich muss überhaupt eine Idee haben wo befinden sich die Schnittstellen auf der Maschine? Da ist so ein Mähdrescher nennenswert groß, da kann man also lange suchen wenn man es nicht weiß. Das wäre die eine Geschichte. Ich muss also erstmal überhaupt ein Verständnismodell haben. Dann muss ich mir ein paar Gedanken machen wo kann ich das Zeugs unterbringen? Dann ist ja die nächste Frage in den Schnittstellen: Wer macht was? Denn ich weiß bestimmte Steuergeräteanteile gehen an andere Standorte. Dafür nutze ich diese Interface-Beschreibung natürlich, um denen schon mal mitzuteilen was geht bei euch rein, was geht raus? Die müssen ja auch wieder im Kontext ihr Ding machen und müssen verstehen mit wem sie es zu tun haben. Das ist jetzt bezogen auf diese elektrischen Schnittstellen. Wenn ich das jetzt mal auf das Thema der Landmaschinen nehme, dann hab ich wieder eine ganz andere Problematik, weil dann ist ja die Schnittstelle auch zum Beispiel vorne mein Schneidwerk, dass da vielleicht 15 Meter oder 20 Meter breit ist. Das gesamte Material muss durch die Schnittstelle Häcksler durch, es muss das physikalische Material bewegen, Dinge damit machen und muss es hinten wieder rausschmeißen. Und eventuell das Zeug abfahren. Da ist das Thema Schnittstelle von dem mechanischen oder physikalischen Material eine ganz starke Herausforderung. Es ist also auch wichtig, dass das was über die Schnittstelle beschrieben wird, an die verschiedenen Entwicklungspartner geht. Es ist im Prinzip der Eingangsvektor, damit diese inhaltlich arbeiten können.

Interviewer (Ricarda Schüssler): Das heißt also insbesondere die Beschreibung von Eigenschaften eines Systems oder einer Komponente wären die größten Use Cases.

Interviewee ID3: Das ist je nach Flughöhe. Denn ich das auf dieser Ebene betrachte, dann bin ich noch relativ allgemein unterwegs. Wenn ich jetzt zum Beispiel an das Hardware-Software-Interface denke, was wir für funktionale Sicherheit benötigen, dann muss ich am Ende das Verhalten für dynamische und statische Zustände an der Schnittstelle beschreiben können. An der Stelle fangen die Leute regelmäßig an zu kotzen, weil das sehr viel Arbeit macht und sie nicht wissen was sie da aufschreiben sollen. Aber das ist halt das, was wir Stand der Technik und nicht Stand der Normen und Standards zeigen

müssen: Haben wir da eine Idee von oder glauben wir es nur?
Interviewer (Ricarda Schüssler): Das bringt mich eigentlich schon direkt zur nächsten Frage. Gibt es Herausforderungen mit Schnittstellenbeschreibungen?
Interviewee ID3: Ja, das gibt es durchaus. Der Hauptknackpunkt heute ist, dass wir es überwiegend manuell, also Text- getrieben machen. Und, dass wir zwar über Requirements-Tools und auch über Architektur Tools verfügen, wir aber heute noch nicht in der Lage sind aus den Architekturbeschreibungen automatische Interface-Spezifikationen heraus zu generieren. Da fehlt uns einfach noch ein Stück. Und auch da muss man es noch mal so sehen: ein Unternehmen wie Claas oder wie andere sind sehr heterogenen. Das heißt, es gibt Standorte, die haben durchaus ein sehr hohes Engineering Know-How und haben eine gute kritische Masse an Experten. Andere Unternehmen, wir haben zum Beispiel einen Standort hier in Dänemark, der baut unsere Kameratechnik und ist ehemaliges Startup. Die haben ein ganz anderes Verhältnis zu Normen und Standards als wir das in der Zentralabteilung haben. Deswegen sind sie nicht schlecht, sie sind einfach anders geprägt und haben andere Needs. Die jetzt auf eine gemeinsame Arbeitsweise einzuschwören das ist einfach eine Lebensaufgabe. Das spiegelt sich da wieder. Selbst wenn wir jetzt in der Lage wären im Head Quarter diese Interfacespezifikationen automatisiert zu generieren und auszuleiten usw., dann heißt das noch nicht unbedingt, dass jeder damit viel anfangen kann, oder das verwenden könnte. Es hilft mir am Ende ja nichts, wenn ich die Schnittstelle auf einer Seite ändere. Ich muss das hier auf beiden Seiten der Systemelemente tun und ich glaube, das ist eine der Hauptprobleme, die wir heute haben. Diese Dinger konsistent zu halten über die Zeit.
Interviewer (Ricarda Schüssler): Du meinst konsistent von den Ansätzen selbst her?
Interviewee ID3: Ja, die Konsistenz der Schnittstelleninformationen über alle Entwicklungspartner. Ich kann das ja nochmal aus einer anderen Sicht beleuchten. Vielleicht ist das nochmal spannend. Wenn wir eine Landmaschine betrachten, dann haben wir häufig Entwicklungszyklen, die liegen bei 5 bis 7 Jahren, bis wir zum Beispiel die nächste Generation der Landmaschine draußen haben. Die Landmaschine, die ist gut und gerne 15 bis 20 Jahre operativ auf dem Feld unterwegs. Das heißt so eine Landmaschine, da habe ich vielleicht einen Lebenszyklus von locker 25 oder 30 Jahren. Und wenn wir eine Schnittstelle ändern würden, nachdem die auf dem Feld ist, da muss ich ja weltweit für den Service so eine Schnittstellenänderung irgendwo festhalten. Es geht um die ganze Welt, das muss man also schon mit Sinn und Verstand machen.
Interviewer (Ricarda Schüssler): Ja, auf jeden Fall. Dann würde ich mal weiter gehen. Und zwar soll es jetzt nochmal speziell um den Schnittstellenbeschreibungs-Ansatz an sich gehen. Wie werden Schnittstellen beschrieben oder spezifiziert und wie sehen diese Beschreibungen aus? Du sprachst eben schon davon, dass ihr dies vor allem textbasiert

macht, obwohl ihr eigentlich Modelle als Quellsystem habt, da ihr diese nicht automatisiert rausbekommt. Vielleicht kannst du hier noch ein bisschen zu erzählen, wie dies genau abläuft und wie eine solche Schnittstellenbeschreibung aussieht?

Interviewee ID3: Erstmal wie entstehen Schnittstellenbeschreibungen? Wir fangen normalerweise mit Anforderungen an, kommen über die Anforderungen zu Architekturen und in den Architekturen sehen wir Schnittstellen. Die Schnittstellen, die wir in den Architekturen sehen leiten wir wiederum als derived requirements aus und schreiben die im Prinzip auf dieser Ebene ins System auf. Das sind am Ende die Dinger, die hinterher bei der Integration getestet und integriert werden müssen. So ist normalerweise der Weg. Das heißt Interfacespezifikationen sind heute bei uns Texte. Also Anforderungen abgeleitet aus der Architektur, die textuell festgehalten werden. Bei Netzwerkknoten zum Beispiel sind das CAN oder Ethernet basierte Beschreibungen mit Signalen, Nachrichten und so weiter. Wo wir eine Zuordnung machen, was da vielleicht der Sinn des Signals ist und wie es verwendet wird. Das findet man in so etwas wie CAN-Matrizen wieder. Das ist relativ ähnlich, da haben wir auch im Prinzip so einen Bruch wenn man so will. Wenn man sich so ein CAN-basiertes System zum Beispiel anschaut: Was ist denn jetzt die Quelle der Information des CAN Signals? Ist das die Datenbank, in der die Signale definiert sind? Oder ist das die Schnittstelle, die vorgibt, welche Signal verwendet werden müssen? Für mich ist das so: die Quelle ist für mich der Need, den wir auf der Maschine haben. Aber die Daten, der Abgleich dieser Schnittstelleninformationen in so einem Tool wie Provision funktioniert am Ende andersherum. Denn dazu ist Ding einfach zu fett also es sieht sich als Quelle und nicht als Senke und das ist zum Beispiel ein Spannungsfeld. Eigentlich wäre es technisch gesehen andersrum, aber was den betrieblichen Alltag betrifft, ist das halt die Realität.

Interviewer (Ricarda Schüssler): Okay, das heißt im Prinzip verwendet ihr reine Texte oder Dateien in natürlicher Sprache und leitet diese aus Modellen ab. Das Ganze geschieht dann händisch, wie ich verstanden habe.

Interviewee ID3: Ja, also wir könnten das mit einigen Tools machen. Es wirkt aber nicht konsequent, also das lebt nicht. Das sind alles noch Experimente, würde ich sagen. Das was tatsächlich funktioniert ist: die Schnittstellen aufdecken und die Anforderung dazu aufschreiben. Zur Automatisierung fehlen einfach noch Schritte in den Toolketten.

Interviewer (Ricarda Schüssler): Die Dokumente, die dann generiert wurden, sind diese nach einem gewissen Schema aufgebaut oder sind die systematisch aufgebaut oder wird dies auch je nach Usecase relativ flexibel gehandhabt?

Interviewee ID3: Ich würde sagen, dass das häufig noch sehr flexibel und sehr informell gemacht wird. Da, wo es jetzt in Richtung der Datenbanken geht, also dort wo das, was technisch umgesetzt werden soll, auf Nachrichten umschlüsseln, da haben wir schon relativ

klare Standards. Aber was das davor betrifft: Ich glaub, da ist noch viel Verwirrung.
Interviewer (Ricarda Schüssler): Gibt es hier auch viel Verwirrung im Hinblick auf Semantik?
Interviewee ID3: Es gibt beispielsweise bei so einer Landmaschine ein bekanntest Signal, das ist die Dreschtrommeldrehzahl. Für die Mechaniker, die in dieser Welt arbeiten, ist völlig klar, was das ist. Für mich aber jetzt als Elektroingenieur und Datenmensch frage ich mich: Kann man das auch anders beschreiben? Was ist das denn, jetzt abstrakt gesprochen? Das ist eines der Probleme. Es gibt Ingenieure, die sind sehr praxisnah, sehr stark an der Lösung, und ich sag immer lass uns mehr abstrakt sein. Je abstrakter wir sind, umso länger können wir mit unseren Engineering-Artefakten arbeiten. Das können die aber nicht, sie haben das nicht gelernt. Und das ist eines der Themen, das wir haben. Wir kriegen Input von Leuten, die in Lösungen denken und geben das dann imperativ auf ein Schnittstellenabstraktions-Layer. Das würde es, glaube ich, treffen.
Interviewer (Ricarda Schüssler): Okay und auf dem Layer muss man sich dann sicher sein, dass jeder, der diese dieses Layer liest das gleiche darunter versteht. Das wäre dann eventuell die Semantik, die vielleicht nicht ganz eindeutig ist. Spannend! Wir waren jetzt bei der Frage 10 also im Bereich modellbasierte Methoden. Wir haben gerade schon besprochen, dass ihr das weniger nutzt für Schnittstellenbeschreibungen an sich. Als nächster Punkt wäre es interessant, herausrauszufinden ob ihr Schnittstellenkontrakte oder Kontrakt-basierte Methoden nutzt?
Interviewee ID3: Ich will nicht sagen, dass wir das nicht haben und dass wir nicht so arbeiten, aber das passiert eher implizit. Ich gebe mal ein Beispiel. Wenn wir jetzt wieder an ein mechanisches Ding denken und denken da zum Beispiel an einen Mähbalken von einem Mähdrescher. Dann ist dieser Mähbalken natürlich an das Interface angepasst, wo er aufgenommen wird. Die wissen, das muss zusammenspielen, also die wissen ich kann jetzt nicht einen riesigen Mähbalken an ein kleines Ding dran schrauben, das passt einfach nicht. Implizit wissen sie das und haben das auch vielleicht explizit aufgeschrieben, aber das würde ich erstmal nicht erwarten. Aber wo ist das interessant? Es ist eigentlich interessant für mich bei dem Thema: Wie entwickle ich denn mein System? Wenn ich also einen top down Ansatz fahre, muss ich von oben nach unten alles neu entwickeln, muss ich alles neu durchdeklinieren. Wenn ich das schaffe mit einem Safety element of context zu arbeiten, wie wir das in der Safety kennen, dann mache ich die Annahme: das wird schon passen. Das, was ich da annehme, das, was ich da baue, wird wohl in diesem Kontext hineinpassen. Wenn ich jetzt ein System Element out of context nehmen würde, dann würde das genau so funktionieren, dass ich nämlich einen Kontrakt hätte mit der Gegenpartei und sage „ich unterstelle du kommst mit dieser Art von Laufzeit und dieser Art von Signalqualität Verarbeitungsqualität zurecht“. Dann hätte ich an dieser

Stelle den Vorteil, dass ich Standardkomponenten verwenden könnte und die nicht immer neu machen muss und ich könnte sagen: dieser Sensor, diese Komponente hat folgende Qualitäten die können wir zusichern, wenn du auf der Gegenseite uns zum Beispiel diese Art von Laufzeit bereitstellen kannst und diese und jene Dinge einhältst, dann passen wir gut zusammen. Und da sind wir vom Engineering her noch nicht.
Interviewer (Ricarda Schüssler): Okay, das wäre gerade im Systems of Systems Kontext extrem hilfreich in diese Richtung zu gehen.
Interviewee ID3: Das werden wir jetzt lernen.
Interviewer (Ricarda Schüssler): Okay, ja, es gibt immer was zu tun!
Interviewee ID3: Es ist leider so. Ich habe in so einem Unternehmen hier oder da sagen wir Olympia-Athleten und ja, man hat aber auch die Dorfklasse mit am Start und dazwischen ist das Spannungsfeld. So ist es halt auch im Engineering.
Interviewer (Ricarda Schüssler): Ich denke nächsten zwei Fragen können wir zusammenfassen. Dort geht es darum, wie diese Schnittstellenbeschreibung erstellt und maintained werden, gerade auch im Hinblick, wenn man Schnittstellenbeschreibungen zwischen unterschiedlichen Filmen hat?
Interviewee ID3: Das ist halt so dieses Gemeine, dass die Schnittstellen nichts sagen. Die tun das einfach nicht, man steckt das Zeug zusammen und man denkt, das funktioniert. Aber man weiß es halt vielleicht nicht ganz so genau, da man das in der Realität vielleicht doch nicht schafft wirklich alle Anwendungsfälle und alle Permutationen zu testen. Und dann klappts halt manchmal nicht, dann stellt man fest, dass manchmal so eine Raketenstufe dann doch vom Himmel fällt, weil es irgendwas nicht geklappt hat. Irgendwann merkt man das, dann hat man das gefunden und stellt das ab. Aber man will das nicht.
Interviewer (Ricarda Schüssler): Okay, genau dann im Hinblick auf Standardisierung von Schnittstellen hast du eben schon einiges zu gesagt. Ich habe verstanden, dass ihr hier eigentlich auch eher weniger macht.
Interviewee ID3: Jein. Man muss jetzt wieder gucken was brauchen wir intern auf unserer Maschine? Und was machen wir nach extern? Also es gibt zum Beispiel auf den Landmaschinen ein standard ISO Bus Interface, den alle Landmaschinen oder viele Landmaschinen verwenden. Das ist im Prinzip genommen eine CAN-Schnittstelle. Das fällt mir natürlich ein und auch andere Standards natürlich. Für uns untereinander ist das nicht ganz so entscheidend. Das ändert sich aber gerade auch mit dem Thema Safety, beziehungsweise Security. Weil wir nämlich nun feststellen, wenn wir keine Standards haben dann haben wir doch ganz schön viel Arbeit. Wir hätten es also sehr viel leichter, wenn wir etwas standardisiert machen würden und haben im Prinzip nur noch einen oder wenige Ansätze und ziehen die Security Betrachtung einmal darüber. Das wäre sehr viel einfacher.

Interviewer (Ricarda Schüssler): Ja, das erspart wahrscheinlich auch einige Rekursionen. Der nächste Punkt wäre: was passiert, wenn so ein Schnittstelle oder eine Schnittstellenbeschreibung sich ändert?
Interviewee ID3: Ja, es ist immer die Frage, ob was hinzukommt oder ob etwas wegfällt. Wenn man eine Abwärtskompatibilität oder Aufwärtskompatibilität hat, wo es am Ende nicht schlimm ist wenn für ältere Maschinen bestimmte Dinge nicht da sind, dann hat man kein Problem. Wenn man aber die Funktional braucht, müssen wir halt sicherstellen, dass dies überall berücksichtigt wird. Wenn man das nochmal jetzt aus Systems-Engineering Sicht betrachtet, würde ich hier sagen, dass da vieles funktioniert, weil die Menschen sich kennen. Es funktioniert eher nicht, weil der Prozess das vorgibt. Da, glaube ich ist es am Ende Papier, da steht irgendwas drin und das kann richtig, kann falsch sein. Ich glaube am Ende ist tatsächlich das Zusammenspiel von Menschen, von Kommunikation und von Menschen, die mit den Systemen zu tun haben das, was uns noch viel hilft.
Interviewer (Ricarda Schüssler): Ja und wo sich vielleicht auch Menschen schwertun ist grade eine interdisziplinäre Schnittstellenbeschreibung aufzusetzen, weil sie eben nicht diese interdisziplinären Sichtweisen innehaben können.
Interviewee ID3: Ja, es reicht halt auch am Ende, dass die Tools, die wir haben, noch nicht so eine Verbreitung, so eine Durchdringung haben, dass man sich auf das, was im Tool steht, verlässt. Das hat einfach noch nicht die Reife, die man dazu bräuchte. Wenn man das jetzt wieder anders vermisst, in Richtung Auswirkungen/ Impact ist es am Ende bei einigen 1000 Landmaschinen, die man im Jahr herstellt, eine ganz andere Nummer als ob man zig Millionen Steuergeräte für Automotive herstellt. Ich kann diese Dinger notfalls nochmal in Sonderinspektion nehmen, im Automotiv nennt man das Teilerückruf. Dann habe ich halt ein paar 1000 Maschinen wo ich etwas machen muss und das kriege ich noch hin. Bei Millionen von Steuergeräten habe ich keine Chance. Das heißt der Schadensausmaß ist in einer anderen Kategorie. Das spielt auch eine Rolle warum man da vielleicht noch nicht weiter ist.
Interviewer (Ricarda Schüssler): Spannender Ansatz. Gibt es Metriken um die Qualität von Schnittstellenbeschreibungen zu messen? Und wenn es das nicht gibt, was würde für dich denn eine gute Schnittstellenbeschreibung bedeuten?
Interviewee ID3: Also Metriken zu Schnittstellen kenne ich nicht, nicht in dem Verständnis. Dass wir ein Architekturelement haben, das auf eine Anforderung mappen und, dass wir gefundene Schnittstellen in der Architektur auch wieder auf eine Anforderung mappen, das haben wir. Dafür haben wir schon KPIs damit da nach Möglichkeit nichts verloren geht. Das kriegen wir hin. Aber ob die Interface-Spezifikation jetzt gut spezifiziert ist im Sinne von: Macht das Spaß die zu lesen, ist die eindeutig geschrieben, ist die verstehbar, ist die auch für einen normierten Englischen nicht Muttersprachler verstehbar oder benutzt

sie ein besonderes Wording? Wobei da würde ich auch wiederum sagen haben wir ein Claas Language Management, das tatsächlich so etwas aufgreift. Wir verwenden intern tatsächlich ein normiertes Englisch, ein normiertes Wording, das quasi eine Claas-weite einheitliche Bedeutung der Worte in verschiedenen Sprachen sicherstellt. Aber wie es hier gemeint als KPI, würde ich sagen leiten wir das heute nicht konsequent aus.
Interviewer (Ricarda Schüssler): Und was wäre für dich eine gute Schnittstellenbeschreibung?
Interviewee ID3: Sie muss übersichtlich sein, sie muss einfach zu lesen sein, muss mit starken Worten arbeiten und am besten auch nochmal verbildlichen, also die Schnittstelle bildlich beschreiben. Sie muss vor allen Dingen aktive und passive Zustände beschreiben. Sie muss das Thema Signalqualität beinhalten. Sie muss auch vielleicht so etwas beinhalten, wie Laufzeit-Dämpfungsverhalten. Wenn ich das aus dem Hardware-Software-Interface heraus zum Beispiel betrachte: ist dieses Signal, das an meinem Interface ankommt, vielleicht schon entprellt? Ist da vielleicht schon eine elektrotechnische Entprellung vorgesehen oder macht der Software-Kollege nochmal intern eine Entprellung und entprellt dann zweimal und es kommt zu einem völlig blöden Ergebnis? Das sind so Aspekte, die würde ich erwarten dort drin zu sehen. Das ist natürlich Arbeit.
Interviewer (Ricarda Schüssler): Ja, das klingt umfangreich.
Interviewer (Martin Becker): Nutzt ihr irgendwelche Standards um so etwas zu beschreiben? Wenn ihr quasi Daten-Qualitäten habt um dann zu charakterisieren: haben wir hier vorverarbeitetes Signal? Oder auch im Hinblick auf Genauigkeit oder Latenzen? Nutzt ihr da einen Standard-Rahmen um die Qualität zu beschreiben oder ist das eher implizit?
Interviewee ID3: Nein, das geht mehr implizit über Kenngrößen wie wir sie für Automotive-Spice oder Agriculture haben. Hat es also ein ASIL oder bei uns ist das ein AgPL- Level. Dann können wir über diesen AgPL-Level auf die Qualität schließen, die dieses Signal haben muss. Es ist aber noch nicht wirklich schärfer. Man könnte das machen und eigentlich muss man das machen. Wenn du zum Beispiel, das Thema Hardware-Software-Interface-Spezifikation nimmst und sagst den Leuten, die die Platinen bauen: Habt ihr denn mal gemessen, ob der Controller an dem Port-Pin genau das macht, was er machen soll? Das wird dir nie einer sagen können, dass sie das für jede Situation und für jeden Mode in dem das Teil geschaltet ist, einmal durch geschippert haben. Vielleicht gibt es einige Nerds, die das machen aber standardmäßig finden sie das immer komisch. Allerdings weiß ich warum ich das frage. Ich habe die Fälle gesehen, wo wir das nicht hinbekommen haben. Da haben wir SPCs gehabt, die wollten wir in einen bestimmten Modus schalten. Und alle waren der Überzeugung, das ist richtig. Es hat aber nie einer nachgemessen und der war nie in dem Modus, wie man gedacht hatte, dass er gehen soll.

Das haben wir auch durch Testen nie rausgefunden, weil wir diese Schnittstellentests nicht explizit gemacht haben. Wir haben die implizit über einen Systemtest gemacht, der Systemtest guckt aber nicht im Bereich Millisekunden, sondern im Bereich Sekunden, daher kann er das gar nicht finden. Das ist ein systematischer Fehler durch ein falsches Engineering-Vorgehen.

Interviewer (Martin Becker): Ja klar, dort muss dann auch anders herangehen an die Stelle. Das war natürlich auch für uns ein bisschen der Hintergrund überall dort wo wir in Richtung Formalismen gedacht haben. Bei den Schnittstellen-Beschreibungen ist je mehr du da reinsteckst, desto mehr könntest du ja vielleicht auch automatisieren oder eine Schnittstelle nutzen. Von daher insbesondere wenn wir jetzt im Systems of Systems Kontext sind und wir vielleicht nicht das haben was ihr habt, nämlich ein implizit gutes Verständnis über diese Schnittstellen, dann müsste ja viel mehr an den Punkten, wo es ein Risiko gibt, dass etwas auseinanderläuft, gegensteuern. Dies indem du näher spezifizierst, was da eigentlich gemeint ist. Dann stellt sich die Frage wie macht man das am besten?

Interviewee ID3: Ich glaube, der Punkt ist am Ende der: Was wir in der Vergangenheit gemacht haben, was insbesondere die Amerikaner viel gemacht haben: wir machen Checklisten. Meterlange, kilometerlange Checklisten. Das Reflexartige, wenn man so eine Checkliste hat, ist „ach schon wieder eine Checkliste ich mach da mal einen Haken dran und dann ist das fertig“. Das ist so die alte Denkweise und deswegen sagt man: irgendwann es hilft nicht immer längere Checklisten zu haben, die ich manuell abarbeiten muss. Wenn du aber mit Automatismen und mit KI arbeiten kannst, mit Maschinen arbeiten kannst, ist die Aussage eine ganz andere. Dann hilft dir nämlich der Inhalt, das Wissen um diese Fragestellung durchaus, wenn das System sich die Antwort selber geben kann. Ich glaube, das ist der Punkt, wo wir hingehen werden. Der Mensch als solches mit der Arbeitsbelastung, die wir heute sehen ist glaube ich in vielen Bereichen einfach am Limit. Das heißt, größere Listen bringen dir keine größere Qualität, weil man am Ende wahrscheinlich nicht mehr als eine bestimmte Menge an Zeit für die Aufgabe verwenden kann. Aber wenn ich das Wissen, das ich sowieso habe, in die Intelligenz stecken kann, ins System stecken kann, dann sieht das ganz anders aus. Dann bin ich eigentlich da, dass ich sage: lasst uns doch immer so ein Concept of Operations aufziehen. Dann kann ich ja immer wieder über meine Systemteile ein Kontextdiagramm machen und werde von oben nach unten, immer wieder kleiner, kleiner, kleiner. Dann kann ich das leisten, oder ich mache das so wie Ricarda, dass mit mir angedacht hat: wir machen Kontrakte. Und unterhalb dieser Kontraktebene wissen wir, die Sache ist gut. Da sind wir uns sicher. Nach oben hin haben wir Risiko, weil wir es vielleicht noch nicht abgesichert haben, auch das wären ja Wege, wie man sich dann nach vorne bewegen können.

Interviewer (Martin Becker): Das interessante ist ja auch, wenn ich an einer Schnitt-

stelle einen Schnittstellenkontrakt habe, dann kann ich gucken, dass das Verhalten an dieser Schnittstelle einigermaßen passt. Aber oftmals haben wir ja auch so Themen, wo wir ein Zusammenspiel an verschiedenen Schnittstellen haben um dann zu sagen so scheint das irgendwie plausibel zu sein. Zum Beispiel wenn jetzt an einer Stelle etwas rein kommt, dann muss 5 Millisekunden später an einer anderen Stelle was passieren? Und wenn wir dann diese Zusammenhänge auch mal aufschreiben würden, dann hätten wir vielleicht auch eine Chance die klassischen Engineering Fehler rauszufinden. Vielleicht kommt man dahin wenn man irgendwann in der virtuellen Umgebung ist oder auch simulieren kann um diese Qualität zu sichern. Das kann aber nur passieren, wenn diese Sachverhalte aufgeschrieben werden. Wenn sie nur implizit im Architekten-Kopf drin sind, dann kann man natürlich nicht erwarten, dass das eine Maschinerie für einen mitbetrachtet.

Interviewee ID3: Was uns heute fehlt ist dieses letzte Stück im Tooling wie in einem Cameo, das uns erleichtert Schnittstellenspezifikationen direkt im Modell mit wenig Aufwand zu beschreiben. Dann beschreibe ich sie direkt in dem Tool und dann generiere das raus, was auch immer ich da möchte. Und wenn ich das tue, dann könnte ich doch auch eine Simulation auf das Modell loslassen. Dann kann ich auch wieder mit Phasen arbeiten, mit Modes arbeiten, mit States arbeiten, mit Use-Cases arbeiten und kann diese Dinge der Reihe nach von oben nach unten runter knacken. Aber dazu fehlt mir noch das Stück für die Automatisierung. Da sind wir noch nicht an der Stelle, wo wir hin müssen.

Interviewer (Martin Becker): Das können wir ja mal mitnehmen. Deshalb mal deshalb machen wir diese Übung um herauszufinden, was da gegebenenfalls noch hilfreich wäre.

Interviewee ID3: Ja, also meine Sichtweise heute. Vielleicht gibt es Toolings heute, die ich noch nicht kennengelernt habe.

Interviewer (Ricarda Schüssler): Vielen Dank! Das hat meine Fragen zum letzten Fragenteil schon gut beantwortet: Welches Verbesserungspotenzial siehst du bzw. was fehlt mit den heutigen Ansätzen was Schnittstellenbeschreibungen angeht? Meine letzte Frage an dich wäre daher: Hättest du andere Fragen in dem Zusammenhang gestellt bzw. was würde dich abseits meiner Fragestellungen sonst noch interessieren?

Interviewee ID3: Man könnte ja mal fragen warum die Interfaces immer die Stiefmütter, oder nehmen wir die Stiefkinder sind? Warum will da eigentlich keiner etwas mit zu tun haben? Es gibt ja eine Sichtweise auf Interfaces, zum Beispiel aus der FMEA. Ich habe immer entweder Energie, Materie, Kraft oder so etwas und ich habe Zustände, die übertragen werden können. Also bin ich bei vier Dingen, die ich über eine Schnittstelle geben kann. Am Ende kann ich sagen, das sind alles Sonderformen von Energie, die da übertragen werden. Wenn ich das jetzt aus Sicht des Systems sehe ist das eigentlich relativ klar. Warum tun wir uns so schwer damit? Das ist nicht klar, aber das könnte der Punkt wiederum sein, dass am Ende an dieser Stelle an der Schnittstelle nämlich das implizite

explizit werden muss. Das Implizite habe ich ja im Kopf. Ich weiß ja, wie das geht aber ich bringe es dem System nicht bei, weil ich das für sinnlos halte. Was aber schon mal helfen kann ist zum Beispiel, wenn man darüber denkt, Anforderungen zu schreiben. Ein gutes Training für Anforderungsschreiber kann sein mal eine Zeit lang als Tester zu arbeiten und mal zu gucken, was die Leute so aufgeschrieben haben. Das verändert den Blick und die Wahrnehmung ungemein. Das könnte man auch hier nochmal überlegen. Wie kriegt man es besser hin im Prinzip die Bedeutung von Interfaces stärker in den Vordergrund zu bringen. Hier auf meinem Schreibtisch habe ich ein Kabel mit einem USB dran liegen. Das Interface seh ich aber nie. Ich sehe im Prinzip nur, da ist ein USB Kabel mit so einem Knopf mit so einem Ding darin so ein Stick, aber was da passiert ist ja von meinen Augen komplett verborgen. Vielleicht ist das somit eines der Gründe was man da machen kann.
Interviewer (Martin Becker): Schön, dass du das so sagst. Der Punkt könnte ja sein, angenommen wir würden hingehen und würden den Leuten sichtbar machen können, was bei den Interfaces passiert. Beispielsweise dass man sagt da fließt pro Sekunde so und so viel Material drüber oder da haben wir ein Protokoll. Es ruft der eine den auf und dann kommt zehn mal das und dann kommt das und ab und zu kommt hier nochmal die Antwort zurück. Wenn man da an diesen Schnittstellen vielleicht besser sichtbar machen könnte was darüber geht, dann würden die Schnittstellen vielleicht auch an Bedeutung gewinnen. Wenn aber nur der Punkt ist wir schreiben jetzt das hier mal auf, dass keiner vergisst, dass wir eine Schnittstelle haben oder dass wir wissen, dass wir miteinander reden müssen, dann ist das ja ein Ding, aber wenn man irgendwie sehen würde und man da ein bisschen mehr beschrieben hat, dann könnten wir ja zum Beispiel Integrationstest daraus ableiten oder Test-Stubs, die ich mal so auf das Ding auf der einen Seite des USBs drauf hängen lassen kann und kann dann mal gucken, was da passiert. Oder man hängt da eine Prüfspitze rein und der sagt meine Annahme waren, dass wir 10 mal pro Sekunde aufgerufen, aber wir hatten den Fall, da hat er das 80 mal aufgerufen. Solche Dinge denke ich, wenn man da mehr leben reinbringen würde, dann wäre das vielleicht auch nicht mehr das Ding, was man sagen würde, es reicht doch in der Art und Weise, wie wir das jetzt beschrieben haben. Also wenn man dort mehr Nutzen erfahren würde, dann wären wir vielleicht auch bereit, das besser zu beschreiben. Eine andere Geschichte, die ich mitgenommen habe ist, dass ich glaube wenn wir auch mal gute Interface Beschreibungen hätte, also eine Art Katalog von guten Interfacebeschreibungen, wo man sieht der Teilnehmer liest Material drüber oder da hab ich irgendwie eine Energiekopplung oder wie hab ich ein bestimmtes Software Protokoll, dass man da vielleicht mal sieht ich könnte noch, dass das oder das beschreiben. Ich glaub das wird es vielleicht auch einfacher machen.
Interviewee ID3: Da hast du recht. Ich glaube, was uns da tatsächlich immer im Weg steht, ist, dass wir nicht abstrakt genug denken. Denn wenn ich die Schnittstelle abstrakt

denke dann ist es doch eigentlich gar nicht so schwer. Wenn ich jetzt aber konkret bin mit der Hand am Arm, dann nimmt der Mechaniker die Schieblehre und misst nach, hat eine Zeichnung, fragt sich was habe ich in der Realität und ist damit fertig. Wenn wir das in Software versuchen, stellen wir fest Mist, geht nicht. Die Schieblehre bekomme ich da gar nicht dran und außerdem ich hab so viele Dinger da drin, da brauch ich ein Tool für. Da merken wir auch den Unterschied am Ende zwischen mechanischer Welt und elektrischer Welt. Aber für mich stimmt das auch am Ende nicht wirklich und da bin ich auch mit den Mechanikern durchaus immer wieder mal im Clinch, weil ich sage ihr müsst doch eigentlich, wenn ihr da vorne so ein Mähwerk dran macht oder sowas auch die Schnittstelle für dynamische und statische Zustände betrachten. Weil ihr werdet doch Schwingungen auf euer System haben, das ist ja Quatsch zu sagen, das muss statisch solche Kräfte aushalten, das muss es ja auch dynamisch können und eigentlich müsstet ihr doch euer mechanisches System auch dafür simulieren können. Also wir sind doch nicht die einzigen in der elektrischen Welt, die mit verschiedenen Modes und Phasen und so weiterarbeiten. Das ist am Ende doch für die Hydraulik und für die Energieversorgung usw. doch genau das gleiche. Es ist nur so, dass die das über ihr Erfahrungswissen nicht mehr im Detail machen müssen. Sie sagen schon, wir haben da so viel Erfahrung mit, das passt. Aber eigentlich, wenn du dir die DIN-Normen einer Schraube durchliest, dann ist das nicht nur die statische, da gibt es auch eine dynamische Betrachtung drin und nichts anderes müssen wir an den elektrischen Schnittstellen machen. Da zwingen uns die Standards zu. Und da wird es dann bei den Mechanikern häufig ein bisschen dünn. Wenn ich so einen Verbund von hydraulischen Steuerdingern da habe, dann habe ich doch die gleichen Effekte am Ende wie in der Elektrik. Ich schalte das Ding ein, dann habe ich gleich eine Druckschwankung oder eine Druckspitze dann sind viele Geräte dran, dann geht das runter habe ich dies hab ich das. Am Ende habe ich die gleichen Probleme, die wir in der Elektrik haben nur vielleicht nicht so viele und nicht mit so einer aggressiven Innovationen wie wir das haben. Ich kann in dem Zusammenhang das Buch von Paul Davies empfehlen, er hat einen Beginners Guide zu Schnittstellen geschrieben. Hat euch das Interview etwas gebracht?

Interviewer (Ricarda Schüssler): Ja, auf jeden Fall! Vielen Lieben Dank für die Antworten und Einblicke. Für mich aus reiner Automobil-Sicht war es extrem spannend, wie das in einer anderen Domäne gelebt wird. Komplett anders ist die Domäne ja nicht und es gibt Parallelen, aber die Arbeit oder die Ansätze sind dann doch unterschiedlich. Daher danke für die interessanten Einblicke!

A.2.3.4. Expert Interview with Interviewee ID4, Automotive Domain

The interview was declared as *Interview ID4*. It was hold on 22.02.2023 via MS Teams in

German. The participants were Interviewee ID4 and Ricarda Schüssler as the interviewer. The following is the interview's transcript in a condensed format:

Interviewer (Ricarda Schüssler): Hallo und Herzlich Willkommen zum Interview! Ich möchte gerne zunächst den Hintergrund der Arbeit und die Motivation dahinter erklären, bevor wir auf den eigentlichen Fragebogen gehen. [Ricarda Schüssler erklärt MIRO Board-Quick Start Guide] Nun zu den ersten Fragen: Kannst du dich bitte kurz vorstellen?

Interviewee ID4: Ich arbeite bei einem Zulieferer in der Automobilindustrie als System Designer.

Interviewer (Ricarda Schüssler): Dürfen wir den Namen deiner Firma erwähnen?

Interviewee ID4: Nein, bitte nicht.

Interviewer (Ricarda Schüssler): Was ist deine organisatorische Rolle im Unternehmen?

Interviewee ID4: Wie gesagt arbeite ich als System Designer. Da meine Entwicklungskollegen in Indien beheimatet sind und keinen direkten Kundenkontakt haben bzw. die Kommunikation hier auch auf Deutsch abläuft, bin ich die persönliche Schnittstelle bzw. der Ansprechpartner für den Kunden für Softwarebelange und andersrum für die Kollegen aus Indien für Kundenbelange.

Interviewer (Ricarda Schüssler): Wie würdest du eine Schnittstelle definieren?

Interviewee ID4: Für mich wäre eine Schnittstelle eine Interaktion zwischen zwei Komponenten. In meinem Fall wäre das die Interaktion zwischen dem Kunden und den Entwicklungskollegen.

Interviewer (Ricarda Schüssler): Welchen Arten von Schnittstellen begegnest du in deinem Projektalltag?

Interviewee ID4: Außer der soeben beschriebenen Kundenschnittstelle habe ich viel mit Software-Schnittstellen, also SW-SW und HW-SW Schnittstellen zu tun. HW-HW Schnittstellen haben wir eigentlich nicht.

Interviewer (Ricarda Schüssler): Wo nutzt du Schnittstellenbeschreibungen, also in welcher Phase eines Projekts?

Interviewee ID4: Rein im Hinblick auf die organisatorische Schnittstelle, die ich eben erwähnt habe, würde ich die Schnittstellenbeschreibung vor dem Projektstart nutzen.

Interviewer (Ricarda Schüssler): Okay, und eventuell währen des Projekts zur Abstimmung, oder?

Interviewee ID4: Wenn ich rein meine organisatorische Schnittstelle betrachte, würde ich die Spezifikation nur vor Projektbeginn nutzen. Bei Software-Schnittstellen sieht das Ganze natürlich anders aus, diese nutze ich auch während des Projekts.

Interviewer (Ricarda Schüssler): Was wären Use-Cases dieser Schnittstellenbeschreibung? Wie helfen sie bei der Arbeit?

Interviewee ID4: Use Cases wären für mich: eine klare Regelung wer was zu tun hat, wer Ansprechpartner ist -dies sowohl seitens Kunde und Entwicklung, also von beiden Komponenten aus. Diese Beschreibungen sind dahingehend wichtig, dass sie Aufgaben klar definieren und abgrenzen. Ansonsten würde jeder für sich etwas tun und am Ende, beim Zusammenschluss würde nichts zusammenpassen.
Interviewer (Ricarda Schüssler): Gibt es Herausforderungen bei Schnittstellenbeschreibungen bzw. wo siehst du Verbesserungspotenzial?
Interviewee ID4: In dem Zusammenhang wären was Kommunikation angeht evtl. weniger Schnittstellen von Vorteil, wobei das hier eventuell auch nicht ganz passt. Wenn die Schnittstelle nicht richtig definiert ist, also übersetzt ist- ich überlege gerade im Sinne von SW-Schnittstellen, beispielsweise eine Umrechnung von uint8 auf uint16- wenn eine Schnittstelle nicht richtig übersetzt ist, das wäre organisatorisch gesehen über die Sprache, beispielsweise der Übertrag von deutsch zu Englisch, dann kann es hier zu großen Schwierigkeiten kommen. Hier muss also die Schnittstelle in beide Richtungen beides können.
Interviewer (Ricarda Schüssler): Wie sieht so eine Schnittstellenbeschreibung aus? Macht ihr dies modell-basiert oder im Sinne eines Dokuments?
Interviewee ID4: Das ist im Sinne von organisatorischen Schnittstellen schwer zu greifen- prinzipiell handhaben wir das als Dokument bzw. als Interaktionsmodell. Das geht eher in Richtung Vertrag. Die Schnittstelle an sich läuft dann viel über informelle Sprache.
Interviewer (Ricarda Schüssler): Nutzt ihr hier modell-basierte Methoden oder sogar interface contracts [erklärt kurz interface contracts]?
Interviewee ID4: Nein, an der Stelle eigentlich nicht.
Interviewer (Ricarda Schüssler): Wie erstellt und maintained ihr die Schnittstellenbeschreibungen?
Interviewee ID4: Auch rein über Dokumente, die gepflegt und versioniert werden.
Interviewer (Ricarda Schüssler): Habt ihr hier Unterschiede, wenn diese in unterschiedlichen Unternehmen erstellt werden?
Interviewee ID4: Grundsätzlich auch über die Dokumente bzw. Verträge.
Interviewer (Ricarda Schüssler): Wie werden Schnittstellenbeschreibungen standardisiert?
Interviewee ID4: Wir verwenden hier das gleiche template, also template-basiert. Ansonsten standardisieren wir hier eher weniger würde ich sagen.
Interviewer (Ricarda Schüssler): Was ändert sich bei Schnittstellenänderungen oder gar, wenn Schnittstellen entfernt werden?
Interviewee ID4: Das haben wir kürzlich bei einem Beispiel miterleben können. Dies führt zu deutlichen Verlusten, leider. Wenn man hier etwas ändert, kommt es bzw. könnte es also zu Problemen kommen. Wenn man sie löscht, kommt es auf jeden Fall zu Proble-

men!
Interviewer (Ricarda Schüssler): Gibt es Metriken, um die Qualität von Schnittstellenbeschreibungen zu messen? Was wäre für dich eine gute Schnittstellenbeschreibung?
Interviewee ID4: Bugs, die zurückzuführen sind auf Abstimmungsproblemen kann man hier sicherlich erwähnen. Die zählen wir momentan nicht, Bug ist Bug, man könnte es aber tun für Statistiken. Innerhalb von bug defect descriptions wird schon immer angegeben, woran es lag, und die Schnittstelle könnte hier durchaus aus Grund drin stehen. Die Schnittstelle kann ja durchaus versagen. Eine gute Schnittstellenbeschreibung bzw. die Schnittstelle als Person in diesem Fall sollte alles von beiden Seiten wissen und somit keine Anfrage nicht verstehen. Hier also eine doppelte Verneinung. Wenn ich beispielsweise als neuer Kollege einen Zusammenhang vom Kunden erklärt bekomme, ich soll dies dann übersetzen und kenne den Hintergrund des Themas gar nicht, dann fällt mir eine Übersetzung ohne Basis Know-How sehr schwer. Wenn ich hier schon länger dabei bin, scheitert es nicht an meiner Übersetzung. Wenn ich den Hintergrund nicht verstanden habe, kann ich nur lückenhaft übersetzen.
Interviewer (Ricarda Schüssler): Interessanter Ansatz, du sagst also das Interface an sich hat hier eine wichtige Rolle, da es beide Seiten verstehen muss.
Interviewee ID4: ja genau, das sieht man am Beispiel Sprache sehr deutlich. Eine deutsche Anfrage einfach in google translate zu übersetzen, erzeugt mir zwar ein Ergebnis, allerdings fehlt mir hier dann der Hintergrund und die eigentliche Bedeutung des Inhalts.
Interviewer (Ricarda Schüssler): Spannend, hier ist also die Semantik von extremer Bedeutung. Das lässt sich sicherlich auch auf nicht-organisatorische Schnittstellen überragen. Was siehst du als Verbesserungspotenzial im Hinblick auf das Thema?
Interviewee ID4: Regelmäßige Treffen der Software System Designer, um sich zu synchronisieren, noch gezielter austauschen zu können und voneinander zu lernen: was ging bei euch schief, dass es bei mir nicht auch schief geht?
Interviewer (Ricarda Schüssler): Kennst du ansonsten vielversprechende Ansätze im Hinblick auf Schnittstellenbeschreibungen bzw. hättest du weitere Fragen gestellt?
Interviewee ID4: Ich wüsste nicht, der Fragebogen war recht umfangreich. Klar, dies waren nun eventuell Antworten, die anders waren als du diese erwartet hast, da es um die organisatorische Schnittstelle ging.
Interviewer (Ricarda Schüssler): Vielen Dank fürs Interview! Ich denke, hier kann man einige Aspekte aus der menschlichen bzw. Kommunikations-Welt in die abstrakte Welt im Sinne von Systemdenken mit übernehmen. Beispielsweise die Semantik war sicherlich ein interessanter Aspekt.

A.2.3.5. Expert Interview with Interviewee ID5, Industrial Plant Engineering Domain

The interview was declared as *Interview ID5*. It was hold on 08.03.2023 via MS Teams in German. The participants were Interviewee ID5 and Ricarda Schüssler as the interviewer. The following is the interview's transcript in a condensed format:

Interviewer (Ricarda Schüssler): Herzlich Willkommen zum Interview, ich würde vorschlagen, dass wir direkt beginnen. Zum Einstieg möchte ich gern den Hintergrund der Arbeit erklären, mit Hilfe eines MIRO-Boards [Ricarda Schüssler erklärt MIRO Board-Quick Introduction].

Interviewee ID5: Interessant, danke für die Erklärung.

Interviewer (Ricarda Schüssler): Kommen wir doch direkt zu den ersten Fragen. Kannst du dich bitte kurz vorstellen: Was ist deine Position, in welcher Branche arbeitest du, welche Standrads werden hier eventuell verfolgt?

Interviewee ID5: Allgemein zusammengefasst ist meine Aufgabe als Maschinenbauingenieur die Projektleitung von Industrieanlagen von der Planung und der Realisierung bis hin zur Inbetriebnahme. Bei Industrieanlagen geht es ja darum, dass Projekte geplant und betreut werden innerhalb des kompletten Lebenszyklus eines Projekts von der ersten Idee bis hin zur Inbetriebnahme. Wir planen und realisieren hierbei für die eigene Firma und auch für Fremdfirmen komplette Industrieanlagen. Die Tools, die dafür verwendet werden, sind die üblichen Microsoft Produkte (Word, Excel, Microsoft Project), CAD (AutoCAD 2D und 3D, Inventor).

Interviewer (Ricarda Schüssler): Gibt es hierbei einen vorherrschenden Standard nach dem gearbeitet wird?

Interviewee ID5: Im Bereich Anlagenbau gibt es eigentlich keine eigene DIN oder ISO Norm, man hält sich aber natürlich bei allen statischen oder konstruktiven Details an die gängigen DIN-Normen, ob es irgendwelche Schraubenverbindungen sind oder ob es Verbindungen von Stahlkonstruktionen sind. Man muss natürlich auch in den Anlagen die Unfallverhütungsvorschriften einhalten, die BImSchG - Gesetzte (also Bundesimmissionsschutzgesetz), dass man also keine giftigen oder schädlichen Stoff in die Umwelt, in die Luft ins Wasser oder in den Boden abgeben darf. Das sind schon Rahmenbedingungen, die auch mitberücksichtigt werden müssen. Baugenehmigungen sind hier auch ein Thema, Gewerbeaufsicht, Arbeitsschutz als ganz großes Thema. Systeme müssen hier so ausgeführt sein, dass niemand im Betrieb von so einer Anlage zu Schaden kommt. Das fängt banal an bei Treppenstufen, die nach einer DIN-Norm ausgeführt werden müssen, oder auch ein Gehörschutz muss berücksichtigt werden. Anlagen sollen also nicht laut sein, d.h. man muss sie kapseln. Anlagen soll man so bauen, dass ein Mitarbeiter auch ohne Gehörschutz, also ohne persönliche Schutzausrüstung auskommt. Man sagt hier also nicht, okay die

Anlage ist jetzt laut, dann musst du eben einen Gehörschutz tragen, sondern man versucht dies im Vorhinein abzufangen. Die erste Maßnahme ist also die Anlage so zu bauen, dass man mit normaler Kleidung die Anlage betrieben kann.
Interviewer (Ricarda Schüssler): Dürfen wir den Namen der Firma erwähnen?
Interviewee ID5: Bitte nicht.
Interviewer (Ricarda Schüssler): Kannst du nochmal kurz deine organisatorische Rolle beschreiben?
Interviewee ID5: Im Prinzip ist meine Aufgabe eine Projektleitung inklusive der organisatorischen Rolle. Es beinhaltet auch die Zusammenarbeit mit den verschiedenen Projektpartnern eines Projekts in einem Boot zu handlen, dass die Schnittstellen stimmen. Im Prinzip ist das keine reine Beratung oder Projektleitung, sondern eine Detailplanung. Wenn das general layout steht, muss hier auch die Detailabwicklung erledigt werden. Da geht es dann viel mehr um Schnittstellen als bei einer reinen Grundsatzplanung. Die Bezeichnung Anlagenplanung bezieht sich hierbei vor allem auf Industrieanlagen. Ich plane und realisiere also Industrieanlagen von der ersten Idee über die Planung bis hin zur Realisierung bis hin zur Inbetriebnahme. Im Prinzip wie bei einem Architekten, der von der ersten Bleistiftidee bis zum fertigen Haus arbeitet.
Interviewer (Ricarda Schüssler): Ich würde dann gerne auf den inhaltlichen Fragenteil kommen. Wie würdest du eine Schnittstelle definieren? Was ist dies für dich?
Interviewee ID5: Eine Schnittstelle ist für mich eigentlich immer der Punkt, an dem das eine Gewerk endet und das andere Gewerk beginnt. Das ist ganz banal eventuell so zu sehen, bis wohin geht die Mechanik mit allem, was hardwaremäßig interessant ist und wo beginnt die Elektrik. Oder innerhalb der Mechanik: wo ist ein Bauteil zu Ende und wo beginnt das nächste, wie passen diese zwei Bauteile am Ende zusammen. In einer Industrieanlage habe ich ja einen riesigen Bereich, in dem ich erstmal ein Erdreich ausheben muss. Der Tiefbauer muss also zunächst wissen wie tief er ausheben muss, welche Rohre in den Boden kommen. Der Betonbauer muss genau wissen wo muss er sein Fundament hinsetzen und wo muss in diesem Fundament zum Beispiel eine Ankerplatte sein, wo er hinterher seine Stahlkonstruktion draufsetzen kann. Wie muss die Stahlkonstruktion aussehen, dass man eine Maschine draufsetzen kann? Wo hat die Maschine wiederum ihren Verbindungsflansch, wo eine Übergangsrohrleitung passt? Das sind also in Industrieanlagen überall Schnittstellen und jedes Gewerk wird von jemand anders realisiert. Das heißt, dass die Schnittstellen ganz klar definiert sein müssen. Schnittstellen sind also immer da, um eine gesamte Industrieanlage überhaupt realisieren zu können. Ganz banal müssen die Bohrungsabstände einer Platte mit den Bohrungen eines Fundaments zusammenpassen. Schnittstellen sind für mich das tägliche Problem und auch das größte Problem. Hierbei diese zu definieren und diese auch durchzusetzen. Gerade, wenn zehn

verschieden Firmen zusammenkommen und jede Firma nur ein Puzzleteil einer Anlage realisiert, muss einer hier immer den Überblick haben und die Schnittstellen klar festlegen, damit alle Partner zusammenkommen können. Da braucht es extrem viel Kommunikation, um solche Schnittstellen zu definieren.

Interviewer (Ricarda Schüssler): Welchen Arten an Schnittstellen begegnest du in deinen Projekten?

Interviewee ID5: Eigentlich hauptsächlich oder ganz vielen mechanischen Schnittstellen. Wie ich gerade schon gesagt habe, sind das Teile, zum Beispiel Maschinenteile, die zusammenpassen müssen. Oder eben Schnittstellen, die Maschinenteile abkapseln, bspw. dieses oder jenes Aggregat funktioniert nur, wenn ich an der Stelle eine elektrische Sicherheitsmaßnahme vorsehe. Ich habe also eine elektrische oder für den Arbeitssicherheitsschutz einen Schutzzaun, wenn jemand dort rein muss zum Arbeiten und macht den Schutzzaun auf muss die Elektrotechnik dies so verschalten, dass die Anlage sofort stehen bleibt. Wenn ich hier eine schnelllaufende Kugelmühle habe mit vielleicht 100 Tonnen Schwungmasse, wenn die einmal dreht, bleibt die nicht von jetzt auch gleich stehen. Wenn jemand also dort herein möchte, muss diese Schnittstelle so definiert sein, dass derjenige erst den Bereich betreten darf, wenn die Schwungmasse steht. Im Anlagenbau habe ich also viele mechanische Schnittstellen zwischen den einzelnen Gewerken, z.B. Maschinenbau, Stahlbau, Rohrleitungsbau, Entlüftungstechnik, Pneumatik, Hydraulik, aber auch sehr viel greift die Mechanik in Schnittstellen für die Steuerungstechnik, also für die Anlagensteuerung rein. Bezogen auf den Fluss geht es hier vor allem um den Materialfluss.

Interviewer (Ricarda Schüssler): Kannst du eingrenzen wann du Schnittstellenbeschreibungen in einem Systemlebenszyklus verwendest?

Interviewee ID5: Eigentlich kommt ein Projektauftrag von einer ersten Idee. Ein Auftraggeber sagt also, ich habe eine Aufgabenstellung, die ich realisieren möchte. Dann gibt es zunächst eine Vorplanung, da brauche ich eigentlich noch keine Schnittstellen zu definieren, jedenfalls in meinem Fall. Gerade auch in Vorprojekten muss man hier mit den Verfahrens Kollegen oder mit dem Einkauf sprechen oder mit den Produktionsleuten oder Werkmitarbeitern. Hier erklärt man dann die Aufgabenstellung (bspw. „Es soll folgendes Material mit folgenden Eigenschaften produziert werden“). Im Vorprojekt muss man dann mit den Verfahrensleuten, mit den Werkleuten sprechen und diese Schnittstellen zwischen den Gewerken und mit dem Vertrieb ausmachen. Diese Schnittstellen definieren also ein Projekt im Groben. Wenn diese abgehakt sind, kann man ins Detail gehen. Da hat man schon von Anfang an Schnittstellen: welche Firmen sind dort mit im Boot um dieses Projekt zu realisieren mit den Aufgabenstellungen vom Vertrieb, vom Werk und von den Verfahrenstechnikern. Man hat also von Anfang an Schnittstellen abzustimmen. Wenn man in die Realisierung geht, hat man Formen für die Realisierung der Elektrotechnik,

für den Steuerungsbau, für die Lüftungstechnik, für Hydraulik, Pneumatik, Stahlbau, Anlagentechnik und die verschiedenen Maschinenlieferanten. Hier muss man dann schon in der Angebotsphase klar machen, was von A nach B angeboten wird. Von dieser Schnittstelle bis zu dieser Schnittstelle bietest du mir was an? Beispielsweise Maschinen haben Schnittstellen im Sinne von Auflageflächen oder Fundamenten, es gibt die Infrastruktur drumherum (Zuwegung für LKWs oder was auch immer). Hier sind also die Schnittstellen zu realisieren. Wenn es dann in die technische Realisierung geht, dann geht man noch mehr ins Detail. Von der globalen Idee, über die Angebotsfirmen bis zum Detail, wo man dann mit jeder Firma die Schnittstelle individuell abstimmt: wie sieht die Verbindung aus, welche Sicherheitstechnik gibt es,...? Man geht also vom Groben zum Feinen, von der ersten Idee bis hin zur finalen Realisierung. Je mehr man ins Detail geht, desto mehr Partner kommen dabei auch hinzu. Beispielsweise das Controlling kommt hier als kaufmännische Schnittstelle in das technische Projekt irgendwann mit hinzu. Oder die Arbeitssicherheit oder Baugenehmigung kommt später im Detail mit hinzu. Anfänglich hat man evtl. nicht gesehen, dass man Abwasserprobleme bekommen kann, da eine Firma die Maschine XY liefert und eventuell mit Wasser spülen muss. Da dieses Wasser dann kontaminiert ist, muss dieses eventuell aufbereitet werden bevor es in die Umwelt gegeben wird. Für mich lebt eine Schnittstelle also durch das ganze Projekt und wird immer detaillierter.

Interviewer (Ricarda Schüssler): Wie nutzt du die Schnittstellenbeschreibungen im Projekt? Was sind also Use Cases für Schnittstellenbeschreibungen?

Interviewee ID5: Im ersten Fall bildet eine Schnittstelle für mich für den konstruktiven Anteil ab, zwei Teile müssen also mechanisch zusammenpassen. Das heißt man nutzt diese Schnittstellenbeschreibungen, wenn diese definiert sind, um die Schnittstellen immer wieder zu kontrollieren. Wenn man für den Anbieter A eine Schnittstelle definiert hat, muss man diese im Nachgang immer wieder kontrollieren. Dass man also in den Verhandlungsprotokollen definiert, Schnittstelle ist diese und diese. Das passiert textmäßig oder zeichnungsmäßig oder mit beiden Ansätzen. Dies muss dann im Nachgang natürlich kontrolliert werden. Wenn man die Fertigungszeichnung dieses Anbieters A bekommt, muss man auf den Unterlagen nachschauen ob er die Schnittstelle in seinen Konstruktionen so umgesetzt hat. Und, das zieht sich natürlich durch, wenn er diese nicht einhält, kommt es natürlich zu Mehrkosten, die dann nachher wieder bei Regressforderungen auch wieder zum Thema werden können. Bei einer Anlagensteuerung ist dies natürlich schwieriger. Dort hatte ich früher immer das Problem, dass die Programme, die die Steuerungsbauer schrieben, nicht lesen kann. Da muss man dann eben sagen, okay, die Aufgabenstellung ist so und so definiert, dass die und die Funktion- zum Beispiel eine Sicherungstür nur aufgehen darf wenn die Maschine steht. Dies muss man schriftlich fixieren und als Projektleiter kann

ich dies dann nicht in jedem Programmschritt nachvollziehen. Bei der Inbetriebnahme wird dann ein Inbetriebnahme-Test gefahren und alle Funktionen werden überprüft und abgehakt. Schnittstellen sind also immer dafür da, dass Aufgaben oder Schnittstellen von wo bis wo jemand zu arbeiten hat, zu definieren. In der Mechanik ist das eher einfach, in der Software oder Anlagensteuerung ist dies dann natürlich schon komplexer. Wie du jetzt denkst im autonomen Fahren, sind es ja im weitesten Sinne auch Anlagen. Man hat hier beispielsweise ein Förderorgan, das einen Materialstrom in einer Zeit X von A nach B bringen muss. Wenn er 90% der Menge gefördert hat, muss er dann langsamer arbeiten um am Ende mit einer feineren Verwiegungs- oder Dosierungsgeschwindigkeit exakt auf 100% Dosiergenauigkeit zu kommen. Das ist in dem Fall die Aufgabe der Steuerungstechnik. So Abläufe beispielsweise werden in der Inbetriebnahme getestet, dokumentiert und abgehakt, so dass man sagen kann: Schnittstelle passt.

Interviewer (Ricarda Schüssler): Wird momentan so ein Verhalten in einer Schnittstellenbeschreibung beschrieben?

Interviewee ID5: Nein, das steht nur in den vertraglichen Bedingungen. Beispielsweise steht darin: die Aufgabenstellung ist, dass ein Förderorgan zu 90% mit einer schnellen Geschwindigkeit fährt, dann über eine FU geregelt auf eine niedrige Geschwindigkeit, in den sogenannten Feindosierstrom wechselt um eine Genauigkeit von plus minus 0,1% bezogen auf 100% zu erreichen. Diese Daten beziehungsweise Vorgaben werden im Test im Rahmen einer Abnahme überprüft.

Interviewer (Ricarda Schüssler): Wie werden diese Schnittstellen beschrieben? Geschieht dies in natürlicher Sprache?

Interviewee ID5: Ja. Das kannst du natürlich zusätzlich noch in Excel-Tabellen umsetzen oder in einem Fließbild. Es gibt eine Art Fließdiagramm, das beispielsweise besagt, von wo nach wo die und die Menge Grobstrom umgesetzt werden soll: Grobstrom z.B. bis 90%, danach soll Feinstrom bis zur Gewichterreichung 100% plus minus 0,1% Abweichung umgesetzt werden. Im Anlagenbau wird hier aber nicht das Programm an sich durchgegangen, das wäre zu komplex. Es wird hier nur die Schnittstelle an sich geprüft.

Interviewer (Ricarda Schüssler): Das klingt spannend, vor allem, dass dies an einer Schnittstelle definiert wird. Ein Teil des System of Systems-Denken geht ebenso in diese Richtung kontraktbasierter Methoden. Man führt hier also Bedingungen an Schnittstellen ein. Man kann zum Beispiel sagen: Wenn du mir – wie du gesagt hast – zu 90% Material mit der und der Güte zur Verfügung stellst, dann kann ich dir im Gegenzug XY zur Verfügung stellen. Man erstellt quasi eine vertragliche Bedingung an der Schnittstelle, die technischer Natur ist.

Interviewee ID5: Es ist auch so, dass man sagt: was passiert denn wenn diese Definition nicht eingehalten wird? Wenn du sagst ich brauche von der Rohstoffkomponente. A 100%.

100% sind aber von der Gesamtmischung nur 10%. Aber diese 10% von dem Rohstoff A müssen zu 100% in die Gesamtmischung rein. Eine Abweichung von plus minus 1% kann ich akzeptieren, aber wenn es jetzt 2% sind, was passiert dann? Dann muss die Steuerung sagen Stopp, ich bin out of range und ich bin nicht mehr in der Bandbreite in der ich mich bewegen darf. Ich muss also eine Warnmeldung an den Anlagenfahrer geben, der die Steuerung vor Augen hat. Dies geschieht zum Beispiel in Form einer Warnlampe und einer Meldung „Anlage stoppt“. Diese besagt, dass die definierte Qualität nicht erreicht werden kann. Dann muss der Bediener, der eine gewisse Entscheidungsfreiheit hat, entscheiden, ob er dies bei der Komponente akzeptieren kann, dass sie nicht 1% Genauigkeit erreicht hat sondern in dem Fall nur 2%. Hier kann der Bediener über einen Button eine Sonderfreigabe erteilen. Wenn der Bediener aber nur einen Spielraum von 2% hat und die Genauigkeit aus Fehlern und was auch immer bei 3% liegt, dann kann er auf den Button drücken so viel er will, dann kriegt er keine Freigabe und dann steht die Anlage. Die Anlage soll in einem gewissen Genauigkeitsbereich die einzelnen Rohstoffe zusammenstellen, irgendwo hin fördern bzw. an ein Mischaggregat geben. Dies muss also eine gewisse Genauigkeit haben. Wenn dies nicht erreicht wird, kann der Bediener in einem gewissen Rahmen eine Sonderfreigabe erteilen. Wenn sich das aber über einer Grenze bewegt, dann ist es Ausschuss. Hier besagt also die Schnittstelle klar eine Bedingung in der Form: „Wenn das, dann muss das passieren“. Dies wird nicht nur rein sprachlich, sondern auch in einem Fließbild ausgedrückt. Beispielsweise auch in einem Blockdiagramm, in dem man erkennt, von wo nach wo der Rohstoff in welcher Qualität, Menge und Genauigkeit befördert wird. Wenn das nicht erreicht ist, dann steht eine Sonderfreigabe durch Bediener zur Verfügung, dies aber nur in einer prozentualen Angabe von X bis Y Prozent. Wenn die Genauigkeit auch außerhalb dieses Rahmens ist, muss die Charge verworfen werden. Hier muss aber die Anlagentechnik die Chance haben dieses Material welches dann 2 Tonnen, 3 Tonnen oder 5 Tonnen in so einem Mischaggregat umfasst, auszuschleusen. Hier kommt ein Gespräch mit den Produktionsmitarbeitern und den Werkleitern zustande, die entscheiden können diese 4 Tonnen nicht wegzuschmeißen, da sie beispielsweise für eine untergeordnete Qualität verwendet werden kann.

Interviewer (Ricarda Schüssler): Nutzt du Kontrakte auch während der Planung einer Anlage zwischen verschiedenen Zulieferern oder zwischen verschiedenen Einzelkomponenten?

Interviewee ID5: Diese Wenn-Dann-Bedingungen nicht, diese werden in der Regel innerhalb von einem System verwendet- hier ist das dann entweder ein mechanisches Problem oder ein Steuerungsproblem. Wir definieren hier schriftlich eher die zeitliche Schiene. So ein zeitlicher Ablauf in einem Projekt ist ja auch eine Schnittstelle. Hier sagt man: du musst bis dann das und das zur Verfügung stellen, damit der nächste daran

weiter arbeiten kann ist. Das ist prinzipiell eine Zeitschiene mit Meilensteinen. Das kannst du über MS Projekt realisieren. Hier definierst du also: Der Anbieter A muss sein Gewerk bis zum Zeitpunkt X liefern, damit B an der Stelle auch weiterarbeiten kann. Das macht man im Kontrakt mit Bezug auf MS Projekt, dass man einen kompletten Zeitplan hat bis zu welchem Tag und Datum wer was zu liefern hat. Und dann gibt es auch die Negierung: Wenn du das bis zu diesem Zeitpunkt nicht liefern kannst, musst du 14 Tage vorher Bescheid geben, damit wir dem anderen noch die Chance geben, dass wir die Zeitschiene verschieben können. Oder wenn du es einfach laufen lässt, dann ziehen wir dir die entstehenden Kosten ab.

Interviewer (Ricarda Schüssler): Ich würde gerne nochmal zurück auf die Frage 9 gehen. Du hast eben schon mal gesagt, dass Schnittstellen unter anderem über beschrieben werden. Wie sieht so eine Schnittstellenbeschreibung weiterhin aus? Realisiert ihr dies rein über Zeichnungen oder über Modelle, tauscht ihr beispielweise CAD-Modelle aus oder sind das Dokumente?

Interviewee ID5: Im ersten Schritt wird die Schnittstelle schriftlich fixiert. In der schriftlichen Fixierung wird dann aber auch Bezug genommen auf Zeichnungen. Zeichnungen sind mittlerweile jedoch eine alte Kommunikationsschiene, mittlerweile geht es um Zeichnungsaustausch. Im Bereich von reinem Anlagenbau ist es üblich, dass Zeichnungen ausgetauscht werden. Der 2D Bereich ist im Moment noch aktuell, aber es geht immer mehr in Richtung 3D-Modelle (step-Dateien), gerade wenn ich mit Kollegen externer Firmen kommuniziere. In der Planungsphase läuft noch viel über zweidimensionale Kommunikation also 2D-Zeichnungen ab, da ist die Schnittstelle noch nicht so exakt zu definieren. Da kann man beispielsweise sagen: Wenn die Maschine da oben auf der Bühne draufsteht, dann steht die da oben auf 3m, dann ist das ok. Die Maschine ist so lang so breit, die hat unten irgendwelche Flansche, die steht auf der Bühne. Da muss ich aber nicht wissen wo sind die Bohrungen, welche horizontale Lasten oder dynamische Lasten bringt diese Maschine in so eine Stahlkonstruktion. Wenn es aber in die Detailabwicklung geht, dann geht es immer mehr in Richtung schriftlich Fixieren, immer aber mit Bezug auf Zeichnungen. In der Abwicklungsphase läuft dann alles im 3D Bereich ab. AutoCAD 3D oder Inventor sind Programme, mit denen ich selbst viel zu tun habe.

Interviewer (Ricarda Schüssler): Wenn solch ein CAD Modell von einer Maschine verschickt wird enthält diese ja sicherlich extrem viele Schnittstellen. Wird in diesem Modell nochmals gesondert auf die Schnittstellen im Modell eingegangen?

Interviewee ID5: Nein, das muss dann schon schriftlich fixiert werden. Zum Beispiel wird gesagt: eure Schnittstelle ist der Einlauf bzw. Auslauf der Maschine und das wird dann entweder in dem Vertrag oder in der kompletten Kommunikation festgelegt. Man sagt: Ihr habt ja schon die Schnittstelle definiert, bis zum Mischeinlauf und ab dem

Mischerauslauf sind eure Schnittstellen. Mit demjenigen, der dies dann im Detail bei der Firma A, B oder C bearbeitet, erklärt man dann: „In der E-Mail vom so und so vielten haben wir die Schnittstelle definiert. Du erhältst anbei das 3D Modell, dort findest du die Einlaufgröße. Das ist Einlauf mit 250mm Durchmesser mit 8 Schrauben und unten ist das ein Rechteck. Das sind deine Schnittstellen, an die du ranmusst.“ Man macht das also immer doppelt. Man sagt nicht einfach: „Hier ist die Zeichnung, suche dir das heraus, was du brauchst“, sondern man muss das schon noch zusätzlich schriftlich fixieren. Eigentlich ist es ein ständiges hin und her von Kommunikation. Das nimmt immer mehr zu, dass man unheimlich viel kommuniziert aber nicht mehr persönlich, sondern vermehrt nur noch per E-Mail.

Interviewer (Ricarda Schüssler): Das heißt im Prinzip hast du ein Modell und hebst es in natürlicher Sprache im Modell hervor. Wir haben über die Fragen 12 und 13 auch bereits gesprochen, hier geht es um die Erstellung und Verwaltung von Schnittstellenbeschreibungen.

Interviewee ID5: Ja, man hat hier außerdem hat einen Änderungsindex, der Änderung A, B, C oder das die Version 2, 3, 4 erkennbar macht.

Interviewer (Ricarda Schüssler): Gibt es einen Prozess um Schnittstellen zu standardisieren?

Interviewee ID5: Grundsätzlich hat man einen gewissen Standard, dass man zum Beispiel die Elektrotechnik nicht mit der Mechanik oder mit der Hydraulik oder Pneumatik oder sonstigem austauscht. Man hat außerdem standardisierte Lieferanten, bei denen man weiß für den Kunden A, B oder C, wenn das jetzt die gleiche Industrie ist, dass die Siloanlage so und so aussieht, die Filteranlage für die Silos sieht so aus, der Mischer könnte der oder der andere sein. Man hat da schon ein paar Komponenten, die wie ein Puzzle oder wie so ein Lego zusammengestellt werden können. Es gibt dann ein paar Standards, die besagen, dass ein Schüttwinkel in einem Industriebereich immer nur so steil einzuhalten ist. Wenn der Winkel zu flach ist, funktioniert das nicht. Hier muss man immer wieder neue Anfragetexte erstellen (bei meiner ehemaligen Firma war dies so), zum Beispiel für eine Förderanlage. Eine Förderbandanlage sieht eigentlich immer gleich aus. Du hast ein Förderband, das hat eine gewisse Länge und eine gewisse Breite. Sie hat gewisse Komponenten, unter anderem einen Antriebsmotor, sie hat eine Sicherheitstechnik und ist manchmal lang, manchmal kurz, manchmal schräg. Da kann man dann schon auf standardisierte Textbausteine zurückgreifen und für unterschiedliche Projekte verwenden. Es ist am Ende dann natürlich je nach Projekt individuell, beispielsweise ein Förderband ist jetzt länger, das ist breiter, das ist steil, das ist flach. Das Silo ist jetzt größer / kleiner, ich brauche noch zusätzliche Besonderheiten. Aber man schon einen gewissen Standard auf den man immer wieder zurückgreift.

Interviewer (Ricarda Schüssler): Was wäre für dich eine gute Schnittstellenbeschreibung? Wie müsste die aussehen?
Interviewee ID5: Eine gute Schnittstellenbeschreibung sollte eigentlich so sein, dass die beiden Partner, die mit dieser Schnittstelle umgehen, verstehen müssen was gemeint ist. Dass man nicht interpretieren kann, denn sonst interpretiert der eine so und der andere so. Daher waren diese standardisierten Anfragentexte bei meiner ehemaligen Firma teilweise 40 – 50 Seiten stark für eine komplette Anlage. Alle Positionen waren detailliert beschrieben. Es war selbst die Blechdicke beschrieben, die Schweißnaht, die Lackierung beschrieben, das Aussehen jeder einzelnen Komponente war beschrieben. Teilweise wird dies ergänzt durch eine Zeichnung und vielleicht noch mit einer Excel-Liste. Man hat den Stahlbau, man hat den Schweißer, den Beschichter, der die Anlage beschichten muss und jeder greift auf diese Spezifikation zurück. Die Kunst es, dass man dann für diese unterschiedlichen Fachleute das so beschreibt, dass alle 3 oder 4 verstehen was gemeint ist. Nicht, dass der eine sagt „jetzt habe ich das geschweißt, jetzt hat der Kollege seine Farbe oder Beschichtung drauf gebracht und jetzt kommst du an und sagst die Schweißnaht ist nichts?“ „Aber dort steht ja drin es soll dieser Typ von Schweißnaht nach der DIN-Norm in dieser Qualität sein.“ Und da haben viele Firmen dann immer gelächelt und haben angerufen: Ich habe hier 40 Seiten Anfrage bekommen, was soll ich denn damit? Sie sollten es so anbieten wie es darin beschrieben ist und das war schon bei der Anfrage immer ein Problem. Normalerweise kriegen wir eine Anfrage einfacher Natur in der Form „baue mir einen Bunker, 3/4/5 Meter hoch, oben rein, unten raus.“ Hierbei bekomme ich aber ja keine Details mit. Unter dem Bunker muss beispielsweise wieder ein Förderorgan sitzen, das wieder unter den Bunker passt. Also muss ich den Flansch definieren. Ich muss die Lage und Größe vom Flansch definieren und auch die Elektrotechnik muss wissen, wo sie ihren Endschalter setzen kann. Das muss in diesem Bunker schon drin sein. Deshalb müssen die Spezifikationen so sein, dass niemand der Beteiligten interpretieren kann, sondern lediglich nach der Spezifikation arbeiten kann. Das war immer im Nachgang auf der Baustelle und irgendwas hat nicht gepasst, der Fall, dass man das Dokument herausgeholt und gesagt hat: „Hier steht 500mm mal 500mm, 50er Flansch mit 8 Bohrungen. Wie viele Bohrungen sind jetzt da drin? Ja 6, verdammt.“
Interviewer (Ricarda Schüssler): Heißt also ihr nutzt Schnittstellenbeschreibungen auch zum Kontrollieren ob alles richtig ausgeführt wurde?
Interviewee ID5: Ja sie sind eigentlich dazu da um das Leben einfacher zu machen. Denn wenn du die Schnittstellen definierst, weiß jeder bis wohin er arbeiten muss und es kommt zu keinen Überschneidungen. Dass einer sagt, ich habe das jetzt bis dahin gemacht und der andere sagt, das war aber doch meine Aufgabe und am Ende passt es dann auch nicht zusammen.

Interviewer (Ricarda Schüssler): Heißt Schnittstellenbeschreibungen werden auch genutzt um Aufgaben aufzuteilen?
Interviewee ID5: Wie gesagt, bei großen Projekten habe ich teilweise 10 Firmen am Tisch sitzen gehabt und da hat jeder mit irgendjemanden eine Schnittstelle. Und deshalb ist es immer wichtig, dass man im Vorfeld von einem Projekt einmal im Monat eine Projektbesprechung durchführt und wenn es in die engere Phase kommt, dann einmal die Woche. Wenn man vor Ort realisiert macht man dann jeden Tag eine Projektbesprechung. Dabei sitzen dann alle an einem Tisch und da kann jeder was sagen und da kann man auch die Schnittstellen abstimmen. Wenn da einer erkennt, dass er ein zeitliches Problem hat und der andere sagt dann ok dann mach ich erstmal das andere und dann kannst du das noch realisieren und dann kommen wir da dann wieder zusammen zeitlich. Aber das ist lebendig, so ein Zeitplan ist ja auch eine Schnittstelle. Da kannst du mit den 10 Partnern immer ganz eng die Schnittstellen definieren und immer wieder abstimmen.
Interviewer (Ricarda Schüssler): Das heißt Kommunikation ist das A und O bei Schnittstellen?
Interviewee ID5: Kommunikation ist das A und O. Vieles geschieht hier noch verbal aber auch per E-Mail-Verkehr oder wie auch immer per Dokumentenaustausch. Ich finde es immer noch einfacher bei einem Projekt, bei dem eine Anlage gebaut wird und bei dem Leute beteiligt sind, die da händisch daran arbeiten. Wenn man bereits ein Anfrageformular oder Anfragedokument mit 40 Seiten oder noch mehr definiert hat, braucht man sich keine Gedanken mehr zu machen, was hinterher mit der Abrechnung rauskommt. Innerhalb einer Projektbetreuung ist es teilweise einfacher miteinander zu sprechen, also eine verbale Abstimmung.
Interviewer (Ricarda Schüssler): Siehst du hierbei Verbesserungspotenzial im Hinblick auf die Schnittstellenbeschreibung? Was sind sonstige Herausforderungen im Zusammenhang mit Schnittstellen, die du siehst?
Interviewee ID5: Eine große Herausforderung ist es, dass die Schnittstellen von allen Beteiligten im Projekt eingehalten werden. Ich persönlich finde, dass es mittlerweile eine Unart ist Emails herumzuschicken und die Leute mit Dokumenten zuzuschütten und zu sagen: Such dir deine Schnittstelle! Ich finde wichtig, dass eine Schnittstelle schriftlich in einem Dokument oder einer Zeichnung fixiert ist. Es gibt im AutoCAD eine Änderungswolke mit der man deutlich machen kann: das ist deine Schnittstelle. Ich habe heute Morgen 28 Fotos von irgendwelchen Zeichnungen bekommen und gesagt bekommen: „Schau mal, ob du damit was anfangen kannst.“ Das ist eine Kommunikation oder eine Abstimmung von Schnittstellen, die du in die Tonne treten kannst. Dabei war ein Verteiler an acht Leute wobei sieben davon nichts damit zu tun haben. Hier muss eine bessere Kultur entstehen, dass man den anderen nicht mit Informationen vollschüttet und sagt

schau mal, sondern das man sich damit auseinandersetzt und prüft welche Schnittstelle den anderen etwas angeht. Man kann ja zum Beispiel in die Zeichnung einen Pfeil oder einen Kreis drum herum machen. Oder es gibt wie gesagt eine Änderungswolke um zu zeigen schau dir das mal an, da gibt es noch ein Problem in unserer Schnittstelle.
Interviewer (Ricarda Schüssler): Das heißt du wünschst dir prinzipiell eine zentrale Abstimmung in einem Dokument oder Modell, sodass nicht etliche Fotos oder Zeichnungen durch die Weltgeschichte versendet werden?
Interviewee ID5: Richtig, diese Mentalität „einfach nur versenden, ich habe mal irgendetwas gemacht“, das reicht nicht für die Definition von Schnittstellen.
Interviewer (Ricarda Schüssler): Kennst du außerdem spannende Ansätze im Bereich Schnittstellenbeschreibungen? Hättest du noch weitere Fragen in dem Zusammenhang gestellt?
Interviewee ID5: Spannende andere Definitionen kenne ich nicht, da bin ich eher so ein bisschen Old-Style unterwegs, nicht so modern. Ich finde es schon ganz gut, wie das mittlerweile mit dem AutoCAD funktioniert mit den Zeichnungen. Nicht wie früher, dass man nur ein einziges Transparentpapier auf einem Zeichenblock hatte, dass man dann kopieren und irgendwo hinschicken musste. Oder einen Ausschnitt auf ein Fax legen und irgendwie markieren musste. Dass das schon mittlerweile mit den 3D-Modellen funktioniert, man diese einem schicken kann und derjenige macht sie mit seinem System auf versteht sie. Außerdem mag ich diese Videokonferenzen nicht so sehr, denn man kann in einer Videokonferenz mit vielen Leuten nie schauen wie der andere reagiert. Diese menschlichen Reaktionen, Gefühle oder Abneigung kann ich nix mit anfangen oder will ich nicht kriegt man in einem 4, 6 oder 8 Augen Gespräch viel besser hin. Spannende Ansätze im Hinblick auf Schnittstellenbeschreibungen, wüsste ich jetzt nicht. Wenn es dann neue Arten gibt Schnittstellen zu definieren oder zusammenzuarbeiten wird man schon sagen: ja cool, wieso bin ich da nicht früher draufgekommen.
Interviewer (Ricarda Schüssler): Ja, ich glaub das Thema Schnittstellen an sich ist extrem wichtig aber die viele denken in ihrem System und bis zu ihrer Grenze. Die Schnittstelle wird dann meist relativ stiefmütterlich behandelt. Vielleicht müssten Schnittstellen in einer Art und Weise höher angesehen werden.
Interviewee ID5: Wenn du ein Projekt machst wie bei mir z.B. im Anlagenbereich oder Industrieanlagenbereich musst du ständig in Schnittstellen denken. Du hast ja nicht nur verschiedene Firmen, sondern auch Behörden und externe Bauämter oder den Umweltschutz, die du mit beachten musst. Es gibt also auch sehr viele gesetzliche Rahmenbedingungen, dies sind auch alles Schnittstellen, die beachtet werden müssen. Ich kenne es eigentlich schon, dass man viele Schnittstellen hat. Es ist aber richtig, dass man zum Beispiel in Firmen, die beispielsweise nur ihren Stahlbau sehen, dass diese sagen:

„Sag mir, was ich machen muss, dann mach ich das. Denen ist das egal was da für eine Maschine draufkommt.“

Interviewer (Ricarda Schüssler): Wir sind am Ende meines Fragenkatalogs angekommen. Wenn du keine weiteren Fragen hast sage ich vielen lieben Dank für das Interview!