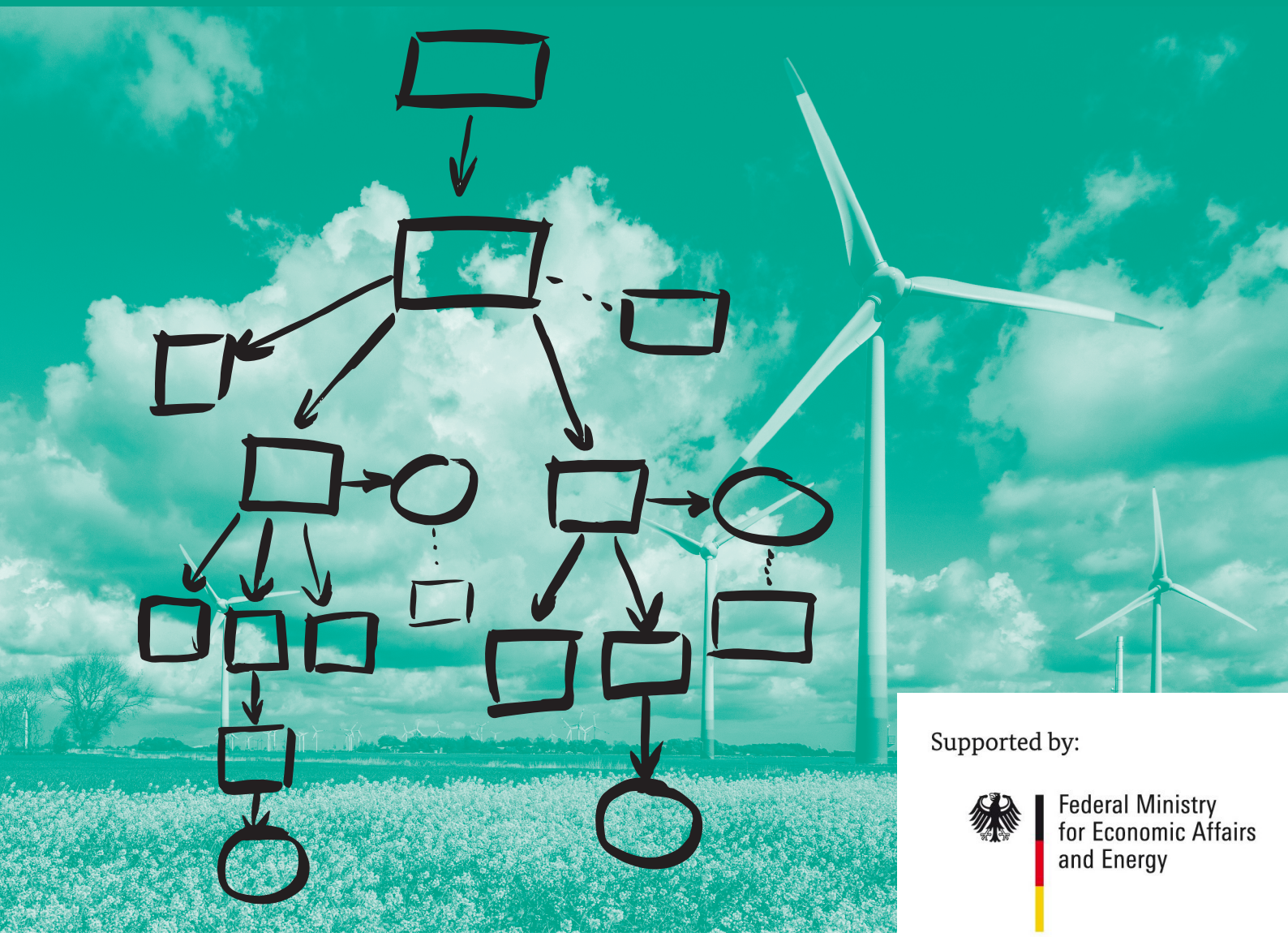
Fraunhofer
IESE

FRAUNHOFER INSTITUTE FOR EXPERIMENTAL SOFTWARE ENGINEERING IESE

ADDRESSING QUALITY PROPERTIES IN USE CASE DESCRIPTIONS

ADDING ASSURANCE TO A USE CASE PROCESS



Supported by:

Federal Ministry
for Economic Affairs
and Energy

on the basis of a decision
by the German Bundestag

FRAUNHOFER VERLAG

Authors

Fraunhofer IESE

Denis Uecker
Dr. Reinhard Schwarz
Dr. Ioannis Sorokos
Hanna AlZughbi

OFFIS e.V.

Dr.-Ing. Mathias Uslar
Sebastian Hanna
Christine Rosinger

ADDRESSING QUALITY PROPERTIES IN USE CASE DESCRIPTIONS

ADDING ASSURANCE TO A USE CASE PROCESS

FRAUNHOFER VERLAG

Contact:

Fraunhofer Institute for
Experimental Software Engineering IESE
Fraunhofer-Platz 1
67663 Kaiserslautern
Germany
Phone +49 631 6800-0
info@iese.fraunhofer.de
www.iese.fraunhofer.de

Supported by:

Federal Ministry
for Economic Affairs
and Energy

on the basis of a decision
by the German Bundestag

Bibliographic information of the German National Library:
The German National Library has listed this publication in its Deutsche Nationalbibliografie; detailed bibliographic data is available on the internet at www.dnb.de.

ISBN 978-3-8396-1724-3

Print and finishing:
RCOM Print GmbH, Würzburg-Rimpar

The book was printed with chlorine- and acid-free paper.

Nobelstrasse 12
70569 Stuttgart
Germany
verlag@fraunhofer.de
www.verlag.fraunhofer.de

is a constituent entity of the Fraunhofer-Gesellschaft, and as such has no separate legal status.

Fraunhofer-Gesellschaft zur Förderung
der angewandten Forschung e.V.
Hansastrasse 27 c
80686 München
Germany
www.fraunhofer.de

Fraunhofer Institute for Experimental Software Engineering IESE

The Fraunhofer Institute for Experimental Software Engineering IESE in Kaiserslautern has been one of the leading research institutes in the area of software and systems engineering methods for 25 years. With its applied research, the institute develops innovative solutions for the design of dependable digital ecosystems. The focus of Fraunhofer IESE is on topics such as “Autonomous Systems”, “Industry 4.0”, “Smart Farming”, and “Smart Energy”, as well as on digital solutions for rural and urban areas. In more than 1,500 customer projects, the institute has transferred cutting-edge research into sustainable business practices, successfully contributing its competencies in the areas of Processes, Architecture, Data, Security, Safety, Requirements Engineering, and User Experience.

Fraunhofer IESE is one of 75 institutes and research units of the Fraunhofer-Gesellschaft. Together, they have a major impact on shaping applied research in Europe and worldwide, and they contribute to Germany’s competitiveness in international markets.

OFFIS e.V.

OFFIS – Institute for Information Technology is a research institute in Oldenburg, Northern Germany. It was founded in 1991 as the first associated Institute of the Carl von Ossietzky University of Oldenburg in the legal form of a registered non-profit organization. From 20 employees in 1992, the team grew to today's figure of more than 250 employees, including graduated computer scientists, physicists, engineers and mathematicians, some with a PhD or with a professorship – with an average age of just 32 years.

Fast knowledge transfer from research to the economy is the crucial foundation for the economic as well as the social well-being of a country. To this end, OFFIS converts scientific know-how from computer science into prototypes, which are then developed further into marketable products by commercial partners.

Energy, Health, Manufacturing, and Transportation – these are the application-oriented research and development divisions that now form the OFFIS structure.

The contents of this publication were prepared in the OFFIS Energy division by the "Standardized Systems Engineering and Assessment" group, which has a focus on developing the use case method (IEC 62559).

List of Abbreviations

ARP	Adress resolution protocol
ASIL	Automotive Safety Integrity Level
AT	Application technique for LSP
BMWi	Bundesministerium für Wirtschaft und Energie (Federal Ministry for Economic Affairs and Energy)
BoK	Body of Knowledge
CAE	Claims Argument Evidence [Framework]
CEN	European Committee for Standardization
CENELEC	European Committee for Electrotechnical Standardization
CM	Criticality Metric
CO	Coordinator
CS	Constituent System(s)
EN	European standard
ETSI	European Telecommunications Standards Institute
FPA	Functional Problem Analysis
GSN	Goal Structuring Notation
GWAC	GridWise Architecture Council
HARA	Hazard Analysis and Risk Assessment
HAZOP	Hazard and Operability Analysis
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
INCOSE	International Council on Systems Engineering
ISO	International Organization for Standardization
IT	Information Technology
LSP	Lego Serious Play
PAS	Publicly Available Specification
QM	Quality Management
QP	Quality Property
RAM	Reference Architecture Model
RAMI 4.0	Reference Architecture Model Industry 4.0
RE	Requirements Engineering
RO	Requirements Owner

SACM	Structured Assurance Case Metamodel
SCIAM	Smart City Infrastructure Architecture Model
SD	System Designer
SGAM	Smart Grid Architecture Model
SINTEG	Schaufenster intelligente Energie
SoS	System(s) of Systems
SoSE	System-of-Systems Engineering
SS	System Support
UC	Use Case
UCMR	Use Case Management Repository
UML	Unified Modeling Language
VIS	Visualization

Table of Contents

Abstract

Our future power grid is probably one of the most complex and most sophisticated critical infrastructures. To model use cases arising in systems engineering, the international standard IEC 62559 proposes a use case template. These use cases are generally employed in combination with a proper architecture model, providing seamless integration in order to trace the requirements from the stakeholders to the actual implementation.

Although commonly used at an international level in smart grid engineering, the emphasis of the IEC 62559 template is clearly on functional use cases rather than non-functional aspects arising in system development. It offers only limited support for critical qualities such as safety or security. However, given the complexity of the smart grid and its safety and security challenges, requirements elicitation for smart grid solutions requires a systematic process to address these crucial non-functional system properties. Use cases should reflect this need.

In this contribution, we propose enhancements to the existing IEC 62559 template and the use case methodology for defining generic smart grid requirements according to IEC 62913. To this end, we show how requirements analysis based on use cases can be suitably embedded into a comprehensive development process specifically devised for systems-of-systems such as the smart grid. In order to meet critical non-functional requirements with the necessary assurance, we propose suitable extensions to the use case template based on assurance cases.

1 Introduction

The research described in this report was motivated by work carried out in the context of the "enera" project [1], a public research project funded by the German Federal Ministry for Economic Affairs and Energy (BMWi) and exploring advanced digitalization concepts for the smart energy grid of the future.

1.1 The enera Project

Enera is one of the five so-called show case projects named "Smart Energy Showcase – Digital Agenda for the Energy Transition" (SINTEG) (in German: "Schaufenster intelligente Energie – Digitale Agenda für die Energiewende") funded by BMWi. From 2017 to 2020, the five SINTEG projects developed transferable model solutions for a secure, economic, and environmentally friendly energy supply with temporarily 100 percent electricity generation from renewable energies. The projects demonstrated their approaches in large-scale model regions[1].

Within the SINTEG program, enera developed and validated solutions for the three main topics *grid*, *market*, and *data* in a model region in the northwest of Lower Saxony, as shown in Figure 1. The work described in this report was carried out within this project in a work package that aimed at the design of the overall enera IT architecture, including an information security concept. This architecture is based upon use cases representing key scenarios in the overall enera project.

1.2 Research Problem

In enera, requirements analysis was based on use cases as a means to capture the needs of the involved stakeholders. From the documented use cases, functional and non-functional system requirements were derived. Alongside binding legal requirements and technical guidelines, these use cases provided a basis for the design specification of the envisioned systems and served as a starting point for risk analysis.

For the modeling of use cases, we employed a use case template according to IEC 62559 [2]. This use case model has been specifically adapted for the needs of the Smart Grid Architecture Model (SGAM), a harmonized European reference des-

[1] See https://www.sinteg.de/en/

ignation framework for the description of smart grids in the electric energy domain. The structure of the use case template reflects the various abstraction and interoperability levels of SGAM.

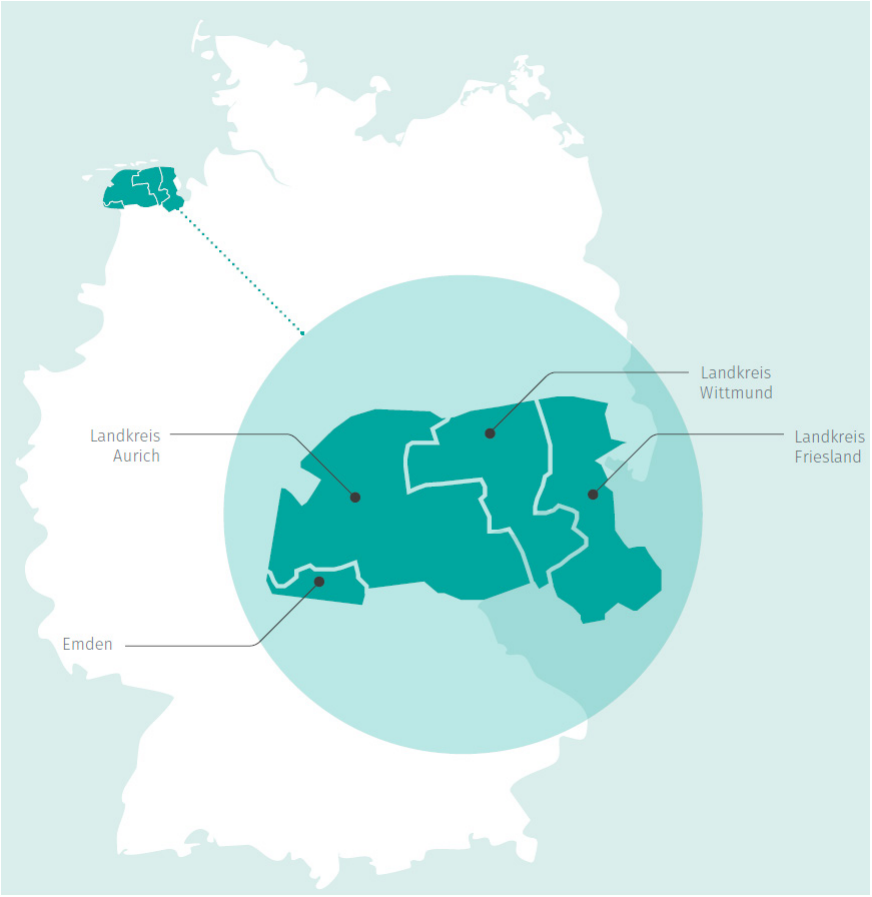


Figure 1 – The model region of enera in the northwest of Lower Saxony, Germany. (Source EWE)

For the specific purposes of the enera project, OFFIS applied some modifications to the original template and managed all use case information in a so-called Use Case Management Repository (UCMR). The UCMR, a tool developed by OFFIS, provides convenient online access to all use cases and offers various capabilities for cross-referencing, electronic search, or export of use case information in different electronic formats.

After the first iteration of use case modeling workshops in the project, it turned out that the project stakeholders had significant difficulties in describing their intended use cases appropriately within the given template structure. Apart from the inherent complexity of Smart Grid architectures, the following problems were observed as the main obstacles by the stakeholders:

- As most stakeholders were experts from the energy domain who lacked proficiency in software engineering, they had problems in choosing an adequate level of abstraction for their use case descriptions. Moreover, they tended to mix product aspects (i.e., aspects regarding the technical system and its operation) and process aspects (i.e., aspects regarding the system's lifecycle).

- Even though the IEC 62559 use case template offers a rich structure content-wise and IEC 62913 [3] provides additional advice on its application, enera participants often struggled with filling in the requested information items in terms of *what* exactly to document and *how* to formulate the template entries.

While these problems were mainly usability issues, use case models raise even more fundamental concerns regarding non-functional requirements, such as safety and security. Because the emphasis of a use case is on how the system should interact with peer systems and the various stakeholders and what the system should do, they typically neglect constraints, that is, what the system must *not* do.

In the enera project, this drawback became apparent when the security and safety risk analysis was carried out: Based on the documented use cases, the risk analysis teams found it difficult to derive suitable safety and security requirements. They found it even more difficult to gain adequate assurance that the design specifications derived from the use cases will, in fact, mitigate the identified risks when the target solution is implemented.

Observing these shortcomings of the use case modeling approach proposed in IEC 62559, OFFIS and IESE aimed at improving the existing standardized use case template and at providing more specific guidelines for the handling of so-called critical requirements. In particular, the focus was on adequate treatment of safety and security qualities, which are key concerns during risk analysis and threat mitigation activities throughout the development lifecycle. More specifically, our main research objectives were to:

- Enhance the current IEC 62559 template to better reflect quality requirements, especially non-functional requirements and the necessary pre- and post-conditions of the use cases to meet these requirements.

- Offer a complementary modeling approach for the assessment of critical system qualities to provide formal assurance that qualities specified in the use case descriptions will be fully covered by the system design and achieved in operation.

- Link the two modeling components – use cases and assurance cases – so that each model cross-references its counterpart.

- Provide methodological guidelines on how to perform the modeling steps systematically to obtain the proposed use case and assurance case models.

In this report, we propose a modeling approach that combines use case models with assurance case models.

1.3 Structure of this Report

In Chapter 2, we argue that the Smart Grid has the characteristic properties of a system of systems. Such complex systems are particularly challenging development targets, and we delineate a process for the systematic elicitation of system requirements for such engineering challenges in the context of smart grids. In support of the proposed process steps, we also suggest some tools and techniques for the elicitation, documentation, and management of requirements and related engineering artifacts. Use case modeling based on the IEC 62559 template is a cornerstone of this framework.

Chapter 3 focuses on quality requirements, especially those requirements that need to be met with a high degree of certainty, such as safety and security properties of the system under development. We introduce assurance cases as an established modeling approach for such critical qualities and describe the foundations of the Goal Structuring Notation as a formal way to document these models.

Chapter 4 explains how our method approaches the analysis and assurance of important system qualities. To this end, the chapter introduces the notion of quality models and shows how a systems and software quality standard such as ISO/IEC 25010 [4] may be employed to identify relevant system requirements, which can then be addressed with corresponding assurance case models as introduced in the preceding chapter. We illustrate this approach with a small example.

In Chapter 5, we explain how use case items and assurance case items are linked for comprehensive mutual cross-referencing and easy traceability between system objectives, between individual steps of use case scenarios, and between interrelated use cases and the corresponding evidence providing quality assurance.

Chapter 6 summarizes the key ingredients of our proposed modeling framework. Based on the insights gained during the enera project, we reflect on their potential strengths and weaknesses.

Finally, Appendix A shows the structure of the IEC 62559 use case template and our proposed extensions. Appendix B refers to the risk assessment metrics used in

ISO 26262-3 [5] as alternative risk metrics to the general ones introduced in Section 4.4. Appendix C provides an example application of our modeling approach for assurance cases that illustrates the concepts described in this report.

2 A Process for the Elicitation of System-of-Systems Requirements

This chapter motivates the need for system-of-systems engineering in the context of smart grids and outlines its implications in the context of the enera project. Key characteristics of smart grids that suggest a system-of-systems approach are the heterogeneity of the systems under study, the many different disciplines of the stakeholders, and the challenging communication required for proper interface design.

First, we will introduce the theory of systems of systems and delineate the corresponding challenges. We will highlight eight core problems that must be addressed in system-of–systems development. To remediate the corresponding pain points, we need to draw on a requirements engineering process that is firmly rooted in system-of-systems theory.

These considerations impose process requirements on our requirements analysis approach, which is based on use case modeling according to IEC 62559. Various other process and artifact requirements emerge as the needs of various stakeholders are considered. Eventually, these factors formed a development method for enera, the so-called Use Case SoSE process, which takes into account SGAM, IEC 62559, creativity techniques, as well as traceable data and requirements management.

During the enera project, it turned out that the standardized IEC 62559 use case template provides only limited support for our envisioned process. This observation inspired several use case extensions that we developed and partially evaluated during the enera project. The proposed use case enhancements will be discussed in subsequent chapters.

2.1 Systems and Systems of Systems – An introduction

Etymologically, the term “system” originates from the Greek “sýstēma”; according to Duden [6], it can be described as a whole composed and structured of separate elements. As versatile as this description can be interpreted, as diverse and context-dependent are the existing definitions of the term “system”.

Many definitions are based on Bertalanffy’s work on General Systems Theory, which attempts to formulate principles and rules that are valid for a large number of systems – more precisely, for all “open” systems. Bertalanffy [7] describes a system as a set of interacting elements and distinguishes between real (i.e., physical) and conceptual systems (e.g., mathematics). Similarly, Boardman and Sauser [8]

describe the essence of a system as "'togetherness', the drawing together of various parts and the relationships they form in order to produce a new whole [...]".

Also based on General Systems Theory, Sillitto et al. analyzed more than 100 current and historical definitions [9] with the goal of defining the hard-to-grasp concept of a system. The results of this work include the definition by the International Council on Systems Engineering (INCOSE) [10], according to which a system is a combination of parts or elements that together show a behavior that is not present for the individual components.

Both Bertalanffy's work and the definition of INCOSE are characterized by a possibly all-encompassing and generally valid understanding of the concept of a system. A potential limitation is the consideration of so-called "Engineered Systems" [11]. Such an engineered system is characterized by the fact that it has been developed with the intention of achieving a specific, previously known goal in a context anticipated during the design process. The overall system can consist of technical (hardware and software), natural (e.g., energy or closed (technical) ecosystems), and social (e.g., stakeholders or users) components, whose interactions generate the actual benefit of the system. Due to the social components, an engineered system can also be considered as a socio-technical system. In the further course of this section, a system is always considered to be an engineered system.

While a system is a collection of parts or elements that together exhibit behavior not present for the individual components, a system of systems (SoS) can, according to ISO/IEC/IEEE 21839 [12], be described as a set of systems that can interact with each other and thereby provide capabilities and functionality that none of the constituent systems (CS) can provide alone. Where in the context of systems one speaks of "elements" or "parts", in the context of SoS we have "constituent systems". This differentiation is important because, as Maier [13] explains, unlike an element of a system, each CS serves its own purpose (Operational Independence) and actually fulfills this purpose independent of the SoS (Managerial Independence).

Analogously to the term "system", an SoS cannot be defined clearly. Boardman and Sauser [14] report that they found more than 40 definitions and approaches for distinguishing between systems and SoS. Among these definitions is the characterization by Maier [13], which serves as a basis for many of the definitions found. Maier names five characteristics of an SoS. In addition to operational independence and managerial independence, which are considered key criteria, there are also the three characteristics of evolutionary development, emergent behavior, and geographical distribution. Boardman and Sauser [14] [15] [16] themselves also list five characteristics for the differentiation of systems and SoS as a result of their literature research: autonomy, belonging, connectivity, diversity, and emergence.

In contrast to the characteristics of an SoS, a widely recognized and now standardized [17] taxonomy has been established for the different types of SoS, which originates from the work of Maier [13] and Dahmann [18]. Maier originally described the three different SoS types “directed”, “collaborative” and “virtual”, which differ in terms of the increasing degree of operational and managerial independence of their CS in ascending order. Dahmann added the “acknowledged” type to this classification, which is essentially a hybrid of the directed and collaborative types.

The ISO standard 21841:2019 [19] defines these four types as follows:

- Directed: The SoS is developed and operated for a specific purpose and by a central authority. The CS are under the control of the central administration and, while in regular operation mode, are subordinated to the SoS and its requirements. Nevertheless, each CS retains the ability to operate independently.

- Acknowledged: The SoS operates under the direction of a management designated by the CS and with the use of dedicated SoS resources. The objectives of the SoS are based on agreements between the CS. The CS maintain their individual and independent development, financing, and responsibility. They are not merely subordinate to the SoS, but also pursue their own intentions and goals. Changes to the systems are implemented in cooperation between the SoS and the CS.

- Collaborative: The CS cooperate voluntarily to fulfill agreed upon purposes. Since there is no central management, the CS decide together on the type of cooperation and compliance with standards. The high degree of autonomy allows the CS to prioritize their own goals and intentions, which can have a direct impact on the quality of the SoS.

- Virtual: An SoS that has neither management nor agreed upon goals or purposes. Completely emergent behavior arises. This emergence can lead to both desired and undesired behavior. The capabilities and maintenance of the SoS depend on “invisible” mechanisms and cannot be determined externally.

2.2 Challenges in Systems and System-of-Systems Engineering

Systems engineering describes a holistic, cross-domain, cross-disciplinary and integrative approach to successfully develop, deploy, and decommission systems [20]. It applies concepts and principles of Systems Thinking as well as scientific, technological, and organizational methods [10].

Traditional engineering disciplines are often component-based and focus on the performance of the component under development. In contrast, systems engineering adopts a holistic view of a problem, including all values and requirements of the stakeholders. The broader scope of system engineering includes not only technical requirements, but also social, regulatory, and economic concerns. Contributions from traditional disciplines (mechanical engineering, electrical engineering, etc.) are evaluated in relation to the whole and integrated in such a way that a coherent overall system is created in which no view of one discipline overshadows the others.

System-of-Systems Engineering (SoSE) finally has an even wider scope than systems engineering. While traditional system engineering is essentially concerned with a single system and its lifecycle, SoSE has to consider a multitude of different and potentially independent systems with their own lifecycles and – depending on the type of SoS – different stakeholders and conflicting interests [21] [22]. The balancing of classical disciplines turns into the integration of the constituent systems (CS). To integrate the CS into a superordinate whole, understanding the CS themselves, their interaction with each other, and how they contribute to fulfilling the purpose of the SoS is essential [22]. Furthermore, changing CS and stakeholder requirements over time must be considered and anticipated [22].

According to the US Department of Defense's "Systems Engineering Guide for Systems of Systems" [22], the underlying architecture on the technical side and the stakeholders on the social-psychological side are of crucial importance for the sustainable success of inherently dynamic and heterogeneous socio-technical SoS. Thus, understanding the CS themselves, their relations and interrelationships, and the motivation of the stakeholders becomes one of the most important aspects. This understanding enables both the identification and assessment of measures to achieve the SoS objectives as well as the evaluation and anticipation of internally or externally driven changes. At the same time, this understanding serves as a basis for creating a technical framework that enables purposeful and user- or stakeholder-oriented further development of the system.

The technical framework can be understood as the architecture of the SoS. The architecture is limited to aspects at the level of the SoS. It leaves details of the CS unconsidered, if possible. Consequently, the SoS architecture mostly consists of the CS and their individual functionalities, the interfaces between the CS, and the information to be exchanged. The requirements of the architecture result from the requirements and the context of the CS and their stakeholders on the one hand, and from the goals, the context, and the environment of the entire SoS on the other hand. Since all CS must integrate within the architecture and adhere to the defined rules for interoperability, the architecture of the SoS serves not only as a "requirement sink" but also as a "requirement source" for the CS.

In the following, different challenges of systems engineering and SoSE will be described, which will then be mapped to corresponding pain points that show the negative impact on practical work within SoS.

Conflicting requirements are not uncommon in cross-domain projects with many different stakeholders. Accordingly, incompatible requirements and resulting tensions between stakeholders occur in SoSE as well. The resolution of these conflicts in favor of the SoSE is usually accompanied by the penalization of one or more CS. According to [22], this disadvantage is only accepted if it is ensured that the affected stakeholders can understand and comprehend the decisions.

Problem 1 *(Understanding). The lack of a shared understanding can lead to unsolvable conflicts, resulting in insufficient solutions or even the failure of the entire project. On the one hand, it is necessary that a shared understanding of the project, its goals, and the abilities of the CS is available across all CS. On the other hand, it is not necessary for each CS to know the motivation or attitudes of every other CS.*

The previously mentioned context and environment of a system play a vital role in the design. Referring to Ackoff [23] and Simon [24], Shah et al. [25] describe the environment of a system as all external elements that can cause changes in the system or establish constraints and boundaries. External elements are those that are not directly involved in the value-adding processes of the system.

Analogously to the emergence of an SoS itself, Shah [25] shows that the context of an SoS can also exhibit emergent characteristics. Both the SoS and the CS can be seen as individual systems in themselves and thus have their own context and environment. By bringing together independent systems, a new context is created, which is neither the intersection nor the union of the previously existing contexts. Rather, the newly created context contains the elements of both contexts that are relevant for the SoS as well as additional, previously non-existent elements that are only relevant for the assessment of the SoS with their respective CS.

Problem 2 *(Context). Both the context of the SoS and the context of the individual CS must be considered during the development and implementation process. Changes in context must be considered throughout the entire lifecycle. Various aspects of the system environment must be reflected in a combined view to obtain a valid overall picture.*

Just as each CS has its own environment and context, different CS and their stakeholders have their own languages, terminologies, and knowledge bases. Combining the expert knowledge of different domains may be seen as a necessity for developing novel solutions that no domain would be able to do alone [26] [27]. However, different perceptions and interpretations of information, tacit knowledge,

and hard-earned expertise can hinder knowledge exchange and communication [27].

According to Shannon and Weaver's "The Mathematical Theory of Communication" [28], Carlile [27] [29] describes these so-called knowledge boundaries on three levels, which arise depending on the degree of innovation and increase in complexity: syntactically, semantically, and pragmatically.

Syntactic boundaries increasingly arise when no common problem description language is established between the parties. Broniatowski and Magee [30] illustrate this boundary with the example of an elevator system, which in British English is often called a "lift", while in American English the term "elevator" is more commonly used. In both cases, the same object is meant. Semantic boundaries arise when there is a common syntax, but the interpretation of terms, information, and situations can vary depending on the particular context. Pragmatic boundaries arise through personal or political interests and the resulting conflicts between parties. These conflicts occur because knowledge from one domain can negatively affect another domain. These negative effects, as well as the effort required to learn something new or transform existing knowledge, reduces the willingness to accept the facts and make the necessary changes. Instead of creating and applying new knowledge by combining different domains, "old", existing knowledge is preferred and a situation that Broniatowski and Magee [30] call a "competency trap" occurs. This competency trap can eventually lead to situations where different domain experts cannot learn from each other, thus preventing the creation of new knowledge.

Problem 3 *(Knowledge Boundaries). Tacit knowledge and knowledge boundaries prevent the exchange of knowledge across domains, the learning of new knowledge, and therefore the creation of new knowledge. However, this new knowledge is required for innovative solutions to new kinds of problems by interdisciplinary teams.*

Syntactic and semantic interoperability are also key factors for the technical implementation of a sustainable and adaptable SoS. Typically, an SoS is initially created with the goal of fulfilling a specific purpose and thus a defined use case. In order to be able to react to new or changing objectives, a reorganization of the CS may become necessary. For CS already in use, newly emerging targets and use cases can mean that they are used in a way or for a purpose not previously intended, which results in new capabilities of the entire SoS (emergence). Sufficient interoperability creates the flexibility necessary for the reorganization of an SoS and allows the recombination of existing CS. The reuse of already existing CS ultimately also leads to increased cost-effectiveness [31].

Van der Veer and Wiles [32] identify four levels of interoperability: technical, syntactical, semantic, and organizational. Each level I_n can only be reached if the previous level I_{n-1} has already been implemented successfully.

- Technical interoperability is achieved once data can be exchanged between systems. Typically, this level involves both individual hardware and software components as well as the infrastructure and protocols necessary for communication between systems.

- Syntactic interoperability generally relates to the data formats being used. It is achieved once the data transmitted in a technically interoperable manner complies with the underlying rules of the selected format. Thus, on this level, besides the well-formed nature of the data, the processing of the data itself also takes place.

- Semantic interoperability is achieved once the previously transmitted and processed data is correctly interpreted and "understood" by all involved systems. While some time ago, human interpretation was emphasized on this level, nowadays machine interpretation of the data has become crucial as well.

- Organizational interoperability addresses the ability of organizations to effectively communicate data and information. This communication must still be sustained effectively, even if, for example, the most diverse information architectures and processes are used or legal framework conditions differ.

The postulated flexibility, effectiveness, and sustainability of an SoS can only be achieved if both the CS and the SoS as a whole are interoperable at least on a semantic, but preferably on an organizational level.

Problem 4 *(Interoperability). The necessary minimum level of interoperability can only be achieved through agreements and resolutions concerning both the technical implementation and the meaning of the content of communicated data.*

While central decisions and concerns – such as financing, goals, or procedures – can be dictated by a central authority in both conventional engineering and directed SoS, the influence of a central administration decreases as the independence of the CS increases (see Section 2.1). At the latest within a collaborative SoS, there is no longer any sole leadership authority.

In a survey conducted by the INCOSE SoS Working Group [33], the lack of central authorities and hierarchical management structures was cited as one of the most important problem areas in SoSE. This led to the question of effective collaboration

patterns in SoS and of the necessary roles, characteristics, and skills for effective leadership in SoS.

Problem 5 *(Leadership). The lack of leadership authority and the issues outlined above generally indicate the need to reach a consensus among all stakeholders involved.*

In order to reach consensus for the SoS and the respective CS and to be able to make decisions that correspond to it, a common understanding of the context and the environment are of crucial importance. The combination of these factors leads to a complex whole, in which elements from various areas interact with each other or exhibit interdependencies. The resulting complexity makes it almost impossible to keep track of the big picture without effective documentation.

In software engineering, inadequate architecture documentation is a well-known problem. Therefore, we could extract the lessons learned from this discipline and transfer them to SoSE in general. Rost et al. [34] were able to confirm the importance of a properly documented architecture in a study. Their survey revealed five main problems, including four that are relevant in this context:

- The documentation of the architecture is usually not up to date and therefore does not provide any real benefit.
- The documentation is provided in an identical way for all stakeholders and tasks, which often means that the information required for a specific task of a specific stakeholder is not accessible.
- The documentation shows inconsistencies in structure, notation or even content.
- The documentation does not provide sufficient options for navigation or search of the required information.

According to Lethbridge [35], two frequently mentioned reasons for outdated or poorly written documentation are the time and effort required to maintain the documentation on the one hand, and the experience that excessive documentation is not exploited to its full extent anyway on the other.

Problem 6 *(Documentation). Lack of effectiveness in use and lack of efficiency in production lead to non-existent, outdated, inconsistent, or incorrect documentation, which in turn leads to high costs for the maintenance and evolution of system architectures.*

Before the actual system design, the most important step for the success of a system is the elicitation of requirements. Underlying methods and processes are summarized under the term Requirements Engineering (RE).

In software engineering, requirements are typically divided into two categories: *Functional requirements* describe what a product should be able to do; *non-functional requirements* usually specify with which quality and under which conditions the required functionality must be provided. Differentiating between these two categories is somewhat misleading, since both aspects must be considered equally to achieve a satisfactory realization.

Apart from system requirements, requirements themselves also have requirements that they should meet [36]. For example, they should be unambiguous, necessary, consistent, and verifiable.

The technical view of a problem as well as the requirements of requirements can usually be met well by traditional SE. In contrast, the more complex and rather holistic problem space of SoSE means that this primarily technical problem analysis is no longer suitable, since, for example, social, organizational, or political dimensions are omitted. Table 1 according to [37] compares the facets, which are of critical importance for RE in SoSE in terms of emergence, ambiguity, and knowledge boundaries.

Table 1 – Differences between Systems Engineering and SoSE from an RE perspective based on [37]

Area	**Systems Engineering**	**SoSE**
Focus	Single complex system	Multiple integrated complex systems
Nature	Technical	Mainly Socio-technical
Objective	Optimization	Satisficing
Expectation	Typical aiming at optimized solution	Initial response and first iteration of a solution
Problem	Well defined	Emergent
Analysis	Technical dominance	Contextual influence and criticality
Goals	Unitary	Pluralistic
Boundary	Fixed and defined	Fluid and ambiguous

The differences from an RE point of view clearly show that the biggest influencing factor is not technical, but rather organizational, political, and social in nature. The interaction of different systems, the goal of satisfying the stakeholders, or even pluralistic objectives and purposes will ultimately be based on human beings as a social component of a socio-technical system. This shows that all characteristics of

an SoS combined lead to the fact that – similar to the Wicked Problems described by Rittel and Webber [38] – there is no single right or wrong solution.

Finally, Keating et al. [37] derived five implications from the differences shown in Table 1, of which four directly affect the requirements of an SoS:

- "The nature of the SoSE problem domain suggests that requirements are simultaneously loose and tight." Requirements should be adaptable to increasing understanding or changing variables. Therefore, they must be as tight as the current situation requires, but also loose enough to be able to adapt to changing circumstances.

- "Requirement resolution should increase with additional understanding of the complex SoS problem domain and emergent conditions." In SoSE, the problem, the approach, and the context are in constant flux. It is therefore important to ensure that the requirements are continuously reviewed and refined.

- "Requirements for the SoS are of a different class than requirements for constituent subsystems being integrated into the SoS." Requirements of the SoS architecture must be of a holistic and integrative nature. Coordination, integration, and management of the SoS should be the focus.

- "Balance must be achieved in the requirements for the SoS." The sole focus on technical requirements and performance inevitably leads to an unsatisfactory and incomplete solution. Requirements must consider the entirety of the problem domain. Besides the technical dimension, this includes the human, social, organizational, and political dimensions already mentioned.

Problem 7 *(Requirements). In contrast to conventional engineering, requirements in SoSE cannot be regarded as immutable and complete. They must be continuously reviewed and refined to reflect the current context and understanding.*

In essence, SoSE is a Wicked Problem, as it must reconcile divergent, sometimes conflicting requirements of different stakeholders, which requires trade-offs and compromises. Accordingly, the SoS problem can only be solved to a certain degree, and most likely no perfect solution exists that would fully satisfy all stakeholders.

Problem 8 *(Wicked). There is no single correct solution for complex socio-technical systems.*

The mapping to the respective Pain Points [33] of SoS in Table 2 shows that the identified problems have a negative impact on the practical work within SoS. Almost every problem contributes to several topics.

Consequently, approaches to solving these problems help to counteract existing challenges of SoSE. The problems themselves can thus be used to derive requirements for the process to be developed in the following chapter. While the process itself has been developed with the goal of supporting the design and implementation of SoS, no fundamental principles can be derived from mere process design. Only the application, further case studies, and long-term experience allow the derivation of such principles. Therefore, the associated column in Table 2 is empty and will not be considered further in the following.

Table 2 – Comparison of the Pain Points [33] of SoS and the identified problems

Pain Points	**Questions**	**Understanding**	**Context**	**Knowledge Boundaries**	**Interoperability**	**Leadership**	**Documentation**	**Requirements**	**Wicked**
SoS Authorities	What are effective cooperation patterns in SoS?	■		■		■			■
Leadership	What are the roles and characteristics of effective SoS leaders?	■	■	■		■			
Constituent Systems	What are effective approaches for the integration of constituent systems?				■		■	■	
Capabilities and Requirements	How can SoSE respond to the SoS capabilities and requirements?	■						■	
Autonomy, Interdependencies, Emergence	How can SoSE address the complexity of dependencies and emergence?	■		■			■	■	■
Testing, Validation and Learning	How can SoSE address SoS validation, testing and continuous learning?						■	■	
SoS Principles	What are the main principles of the SoS mindset?								

2.3 SoSE Process

The challenges described in Section 2.2 strongly indicate that the challenges in SoSE cannot be adequately met by a single process alone. The complex interaction of the CS and their stakeholders within socio-technical SoSE means that each of the problems described must be addressed in various ways and at multiple levels. A significant part of the problems primarily relates to aspects of social interaction or policy constraints of the participating companies and less to technical challenges.

The process developed in this section is meant to be a part of the solution, but not the only solution. Accordingly, the requirements identified below do not address all the problems previously described and do not fully cover those addressed. Furthermore, we assume collaboration and participation willingness for all stakeholders and CS.

To allow the integration of the process into existing process models with as little effort as possible, the structure and description of the process is oriented toward feasible partial models of process models. Broy and Kuhrmann [39] structure process models by using sub-models, which they derive from the question “Who develops what and how do they do it?”. These sub-models are the artifact model (What?), the role model (Who?), and the activity model (How?). If a temporal component is added, the sequence model (When?) forms the fourth sub-model, which describes the sequence of the activities. Across all sub-models, we need to consider the existing constraints, methods, and best practices.

According to the problems identified in the previous section, the requirements for the process are defined first. Based on these requirements, the artifact model is developed and the artifacts necessary to fulfill the requirements are described. The artifact model is part of the foundation of the activity model. The identified activities can generate or provide necessary artifacts, or they can also serve the specified requirements. Subsequently, the roles required to perform these activities are defined in the role model. The chronological order of the activities is defined in the process model, including the artifacts that are involved. Finally, the sub-models are integrated into the overall process.

2.3.1 SoSE Process Requirements

Problem 1 (Understanding) states that a lack of shared understanding can cause unsolvable conflicts with serious consequences for the success of an SoS. These consequences range from failing to meet stakeholder expectations to the failure of the entire project. Such conflicts are not uncommon in interdisciplinary teams with different stakeholders from different domains. Although not every CS has to develop an all-encompassing picture of other CS, knowledge about their goals and

concerns fosters the development of satisfactory solutions. On the other hand, it is essential that each CS understands the purpose, goals, and capabilities of the SoS.

Requirement 1 *(Establish a shared understanding): The activities as well as the artifacts resulting from them shall help to develop and maintain a shared understanding of the purpose, goals, and capabilities of the SoS across all stakeholders.*

The willingness to collaborate, which is necessary for the formation of a shared understanding, also contributes significantly to dealing with a lack of leadership authority, as described in *Problem 5 (Leadership).* In SoS, with increasing independence of the CS, there are fewer and fewer central leadership authorities. From the perspective of the SoS, goal-oriented decisions can only be made collaboratively among stakeholders. Furthermore, collaborative action also plays an important role in the context of the desired interoperability described in *Problem 4 (Interoperability).*

Requirement 2 *(Enable collaboration): The methods used in the process shall facilitate collaboration and trust as well as consensus building among stakeholders.*

Generally, the desired as well as the necessary level of interoperability described in *Problem 4 (Interoperability)* should always be considered in the context of a concrete use case. It is not necessary that all CS of an SoS are interoperable with one another. However, if new use cases for an SoS emerge, suitable CS should be able to achieve sufficient interoperability with little effort [34].

Requirement 3 *(Ensure interoperability): The process shall establish the interoperability necessary for the specified use cases.*

Sensitive information, which the CS keep and need for their economic operation, is often a limiting factor for collaboration. Confidential data can severely limit the information exchange between the corresponding CS. This is particularly relevant if some of the sensitive data must be disclosed to a selected group of stakeholders in the course of joint projects. The stakeholders must therefore be enabled to collaborate without disclosing sensitive and critical information to unauthorized stakeholders.

Requirement 4 *(Ensure confidentiality of information): The tools and methods used in the process shall allow stakeholders to manage sensitive data necessary for collaboration so that it is inaccessible to unauthorized stakeholders.*

Apart from confidential information, a variety of additional information must be provided to obtain a comprehensive and valid overall picture of the SoS for collaboration. Besides the system architecture – which is mainly determined by the CS,

their interfaces, and the relations between the CS – this information is mainly derived from the CS and SoS contexts described in *Problem 2 (Context).* The strong interdependence between the technical architecture and the conditions described by the contexts requires the simultaneous and joint consideration of all information relevant for the considered use cases.

Requirement 5 *(Visualize all relevant information simultaneously): The tools used in the process and the resulting artifacts shall represent all the information and framework conditions necessary for a valid overall picture in an aggregated manner.*

Another major challenge in SoSE is the development of a sufficiently dynamic and adaptable architecture that is capable of representing the constantly changing requirements, contexts, and emerging use cases described in *Problem 7 (Requirements).* Such an architecture is not designed once, but is subject to continuous change.

Requirement 6 *(Enable continuous monitoring): The process shall take into account potentially occurring changes in requirements, context, and environment by providing continuous monitoring.*

It should require little effort to integrate the devised SoSE process into existing models and processes. Both agile and conventional models should be supported. By integrating the solution into established processes, acceptance increases and the perceived additional effort for process implementation is reduced.

Requirement 7 *(Enable integration into existing models): The components of the process shall be integrable into agile as well as into conventional process models.*

The tools supporting the process also affect the acceptance of the solution. Stakeholders in projects are often separated in time and location. To overcome these geographical and temporal boundaries, web-based systems are a common remedy. Unfortunately, most of these systems are general-purpose tools that lack specific support for SoSE artifacts of various kinds. The lack of specialization will often result in the use of makeshift solutions for the exchange of abstract and complex issues such as the system architecture. Such workarounds ultimately aggravate mutual understanding. By using a web-based and at the same time domain-oriented software, geographical and temporal separation can be overcome.

Requirement 8 *(Use purposeful tools and methodologies): Tools and methods used in the process must be domain-oriented and usable in daily business even by separated engineering teams.*

Table 3 summarizes all previously mentioned requirements and assigns them to the identified problems to whose solution their implementation contributes. *Requirement 7 (Enable integration into existing models)* and *Requirement 8 (Use purposeful tools and methodologies)* primarily refer to necessary factors for successful implementation of the process rather than to the identified SoSE problems. Therefore, none of the problems are directly assigned to them.

Table 3 – Assignment of the identified problems to the derived requirements

Requirement	**Understanding**	**Context**	**Knowledge Boundaries**	**Interoperability**	**Leadership**	**Documentation**	**Requirements**	**Wicked**
1. Establish shared understanding	■		■	■	■		■	■
2. Enable collaboration	■	■	■	■	■		■	■
3. Interoperability			■	■				
4. Confidentiality of information						■		
5. Simultaneous visualization of relevant information	■	■	■	■		■		
6. Continuous monitoring	■					■	■	
7. Integration into existing models								
8. Purposeful tools and methodologies								

2.3.2 Artifact Model

The artifact model defines all artifacts of the process model. Artifacts are defined as any results and intermediate outcomes that are created by the activities of the process. The artifact model specifies the content and form of the artifacts. Additionally, it may define dependencies between artifacts [39].

2.3.2.1 Shared Understanding

Shared understanding is the least graspable artifact and is directly derived from *Requirement 1 (Establish a shared understanding).* Referring to Smart et al. [40], Bittner and Leimeister [41] define it as the capability to coordinate the behavior of independent agents of a group towards a common goal. Shared understanding is based on mutual knowledge, beliefs, and assumptions about the task, the group,

the process, or the tools and technologies being used. All of these aspects are potentially subject to change.

The definition clearly shows that shared understanding and the resulting capabilities arise primarily between the people involved in the process. It can be difficult or impossible to record or technically document it to its full extent. According to Aranda [42], a typical documentation usually lacks informal, anecdotal, or only vague information, which, however, is essential in achieving shared understanding. Furthermore, this kind of documentation suffers from the same difficulties as the system documentation described in *Problem 6 (Documentation),* such as inconsistencies, missing updates, and lack of attention. To make matters worse, the results of a process whose success depends on interpersonal and direct communication are recorded in an asynchronous and indirect way.

2.3.2.2 Body of Knowledge

Nonetheless, documenting shared understanding remains an important task to form a common Body of Knowledge (BoK) among all participants. This BoK assumes the function of documentation. In particular, it includes both the architectural design and the architectural description.

However, the BoK must be understood not only as an information sink but also as a development tool. This aspiration also includes looking at it in a less formal way and, for example, supplementing it multimodally with sketches, notes, photos, videos, and so on. Given appropriate tool support, relations between and annotations on the "individual parts" can help to make discussions, thoughts, and results more comprehensible.

A capability to create historical states – similar to a version control system – can also help to make decisions more transparent and to better understand the impact of the influencing factors. To fulfill *Requirement 4 (Ensure confidentiality of information)*, the concrete realization of the BoK must also provide mechanisms for sharing confidential data only with selected stakeholders.

2.3.2.3 Ubiquitous Language

Shared understanding and its documentation have in common that they require a common, ubiquitous language to prevent misunderstandings and ambiguities. In the context of software development, the term ubiquitous language was coined by Eric Evans [43]. It refers to a uniform, problem-oriented language used by all stakeholders. Evans explains the necessity of such a language by the fact that business and technical experts need a common language to communicate unmistakably with each other in all situations. Tenzer and Pudelko [44] also state that language

barriers between different native tongues can have a strong negative effect on the successful formation of a “Shared Mental Model”.

Applied to the heterogeneous group of stakeholders, such a uniform vocabulary helps in cross-domain and interdisciplinary communication. The language contributes to the formation of a shared mental model and thus fosters shared understanding. In addition, this approach promotes thinking in the problem space and from a problem perspective, avoiding the artificial limitation of the solution space by an overly technology-driven approach.

The ubiquitous language itself represents an independent, dynamic artifact that changes with the SoS. It is part of the BoK and should be used in all forms of communication and documents. It can be captured simply by a glossary, a thesaurus, or by something more complex, an ontology. However, its manifestation as a tangible artifact only *supports* the establishment of a ubiquitous language, but does not constitute full realization of this concept.

2.3.2.4 Standards and Standardization Gaps

Another important part of the knowledge base are the standards to be used and established in the domains participating in the SoS. Their terminology, methods, and facts influence both the ubiquitous language and the shared understanding. In addition, standards directly contribute to the interoperability requested in *Requirement 3*. Appropriate standards can ensure the correct transmission and processing of data as well as uniform interpretation of information across all participating systems.

Since the CS of an SoS – specifically of a new use case – usually already exist before the newly arisen use case, it is desirable that both the CS and the SoS consider and implement common standards. This ensures that suitable CS can be empowered to work together with little effort. From a CS point of view, the advantage is that – if the CS itself has implemented applicable standards – no negative effects on local operation caused by required adaptations are to be expected.

Employed or applicable standards can be documented in a simple list. However, such a list offers little added value, so further information should be provided. In addition to the area of application and the domain of the standard, this includes the CS already implementing the standard and information about the use cases in which the standards are used. The same procedure should be followed for identified standardization gaps in order to offer participating CS the option to close these gaps collaboratively and to advance standardization.

2.3.2.5 Summary

In total, four artifacts were derived from the previously identified problems and requirements, which are summarized again in Table 4.

Table 4 – Artifacts summary

Artifact	Description
Shared understanding	Ability to coordinate behavior towards a common goal. This is based on mutual knowledge, beliefs, and assumptions. [41]
Ubiquitous language	A ubiquitous language with uniform vocabulary, understood by all stakeholders and used in all artifacts.
Standards and standardization gaps	Relevant standards and lack of standardization in areas where uniform interfaces and procedures are required.
Body of Knowledge	The BoK assembles all information in a multimodal way and is accessible to all stakeholders. It serves not only as an information sink but should also be understood and used as a development tool.

The relations between the artifacts are summarized by Figure 2. The shared understanding as well as the standards to be used including the identified standardization gaps are part of the documentation. This multimodal documentation can be considered as a project-specific BoK.

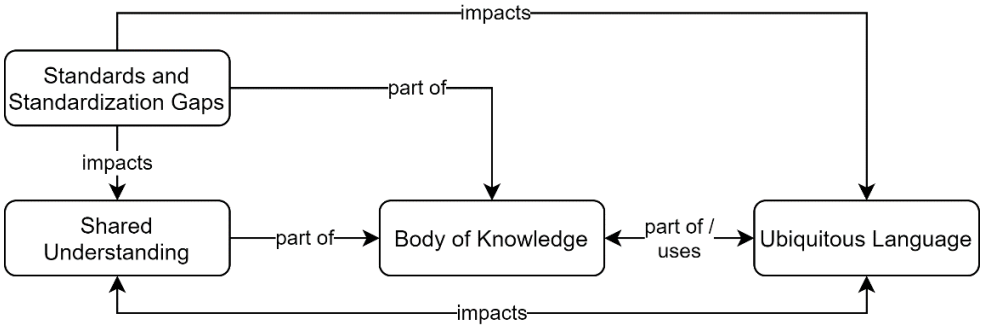


Figure 2 – Artifact relations

Table 5 assigns the derived artifacts to the requirements to which they contribute.

Table 5 – Assignment of requirements to artifacts

Requirement	Shared Understanding	Body of Knowledge	Ubiquitous Language	Standards
1. Establish shared understanding	■	■	■	
2. Enable collaboration	■		■	
3. Interoperability	■			■
4. Confidentiality of information			■	
5. Simultaneous visualization of relevant information			■	
6. Continuous monitoring				
7. Integration into existing models				
8. Purposeful tools and methodologies			■	

2.3.3 Activity Model

The activity model includes all activities that are performed during the process. Completing an activity usually creates a new artifact or an updated version of an existing artifact. The foundation of the elements of the activity model are – similar to Section 2.3.2 (Artifact Model) – the requirements with their underlying challenges, as described earlier in Section 2.2.

2.3.3.1 Establishing a Shared Understanding

Besides the artifacts described above, various activities for the activity model can also be derived from *Requirement 1 (Establish a shared understanding).* At first, the requirement itself can be understood as an activity. However, this coarse-grained activity “Establishing a shared understanding” is subdivided into further sub-activities as shown in Figure 3.

The overarching activity is essentially composed of two sub-activities: “cross-domain discourse” and “externalizing tacit knowledge”. The cross-domain discourse generates the knowledge artifact Ubiquitous Language. The superordinate activity “Establishing a shared understanding” accordingly generates the shared understanding. The cross-domain discourse, the externalization of tacit knowledge, and the fostering of active participation contribute directly to *Requirement 2 (Enable collaboration).* In the following, the sub-activities will be explained in more detail.

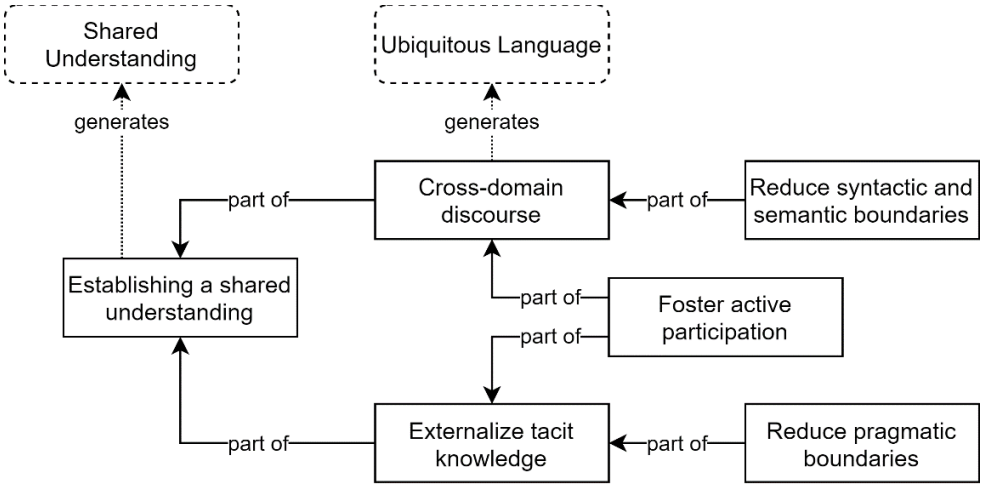


Figure 3 – Sub-activities of the activity "Establish a shared understanding"

Problem 1 (Understanding) underlying *Requirement 1* states, among other things, that the occurrence of conflicts in interdisciplinary teams with different stakeholders from different domains is quite common. However, the different perspectives of the heterogeneous group also provide an opportunity or even necessity for innovative and emergent solutions. Hoffman and Maier [45] argue that heterogeneous groups develop holistic solutions by resolving divergent perspectives and that the discourse to be led contributes to the formation of a shared understanding.

In order to actually bring together the heterogeneous views of the stakeholders in a single solution, it is inevitable to express, listen to, and discuss opinions and perspectives. Hoffman and Maier [45] explain in this respect that satisfaction with a solution depends largely on the satisfaction of the participants with their individual influence on the solution itself. Thus, the necessary discourse leads to a higher acceptance of the solution. Furthermore, this discourse is essential to reducing the syntactic, semantic, and pragmatic knowledge boundaries described in *Problem 3 (Knowledge Boundaries).*

Therefore, cross-domain discourse is an important activity within the process, which increases both the acceptance and the quality of the solution. Since we assumed above that all participating stakeholders want to actively participate and collaborate, it must be ensured within the discourse that, as stipulated in *Requirement 2 (Enable collaboration)*, each participant is given the opportunity to participate and that active participation is encouraged.

Active participation is also essential for uncovering the personal motivation and interests of the individual stakeholders. These two aspects are among the main drivers for overcoming pragmatic knowledge boundaries. Overcoming of these knowledge boundaries is supported, among other things, by concrete insights into

the other domains and CS as well as by the externalization of the stakeholders' tacit knowledge. Since the discourse itself does not necessarily lead to this externalization, it must be explicitly addressed by the methods used in the activity "Externalize Tacit Knowledge".

2.3.3.2 Identifying Standards

The identification of relevant standards contributes considerably to the sustainable establishment of the interoperability specified in *Requirement 3 (Ensure interoperability).* The result of this activity is a collection of standards and standardization gaps captured in the corresponding artifact.

2.3.3.3 Updating the Documentation

The associated documentation should always be kept up-to-date in order to always know the actual state of the project and each design artifact and to make it centrally accessible to all involved parties – for example, to avoid information silos. As already explained in *Problem 6 (Documentation)*, too much effort for maintaining the documentation leads to it being neglected. Therefore, the documentation should also be a tool for discussing and describing new framework conditions or changes in architecture. The added value thus created helps prevent the documentation itself from being abandoned. The result of this activity is an updated version of the BoK.

2.3.3.4 Monitoring Framework Conditions

Updating the documentation requires that the changed framework conditions, their effects, and the implications for the architecture are monitored and verified in an upstream activity. This activity also partially satisfies *Requirement 6 (Enable continuous monitoring).* As a result, the identified changes provide input for both architectural changes and for updating the documentation. Thus, the updated BoK represents the result of this activity.

2.3.3.5 Summary

A total of four activities were derived from the previously identified problems and requirements, which are summarized in Table 6.

Table 6 – Activities summary

Activity	Description
Establish a shared understanding	The establishment of a shared understanding is divided into the sub-activities externalizing tacit knowledge and cross-domain discourse. Both sub-activities require active participation and aim at overcoming knowledge boundaries.
Identify standards	Standards are a cornerstone for manufacturer-independent interoperability. It is necessary to identify the standards and standardization gaps that are relevant for the SoS and its use cases.
Monitor framework conditions	Conditions, context, and requirements of an SoS are subject to constant change. Therefore, these aspects are monitored, and changes are identified.
Update documentation	Relevant information is recorded in the project-specific BoK. To prevent this information from becoming obsolete, an explicit activity for updating is integrated into the process.

Figure 4 shows the relations while Table 7 assigns the activities to the identified requirements.

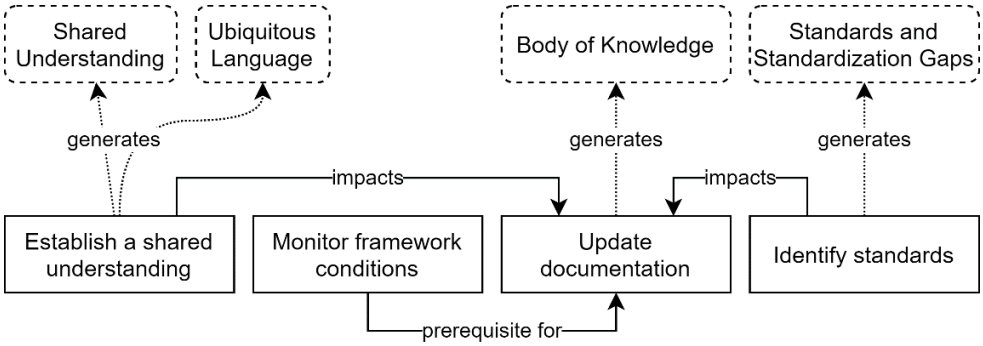


Figure 4 – Activity relations

Table 7 – Assignment of requirements to activities

Requirement	Establish a common understanding	Identify standards	Monitor conditions	Update documentation
1. Establish a shared understanding	■		■	
2. Enable collaboration	■			
3. Interoperability		■		
4. Confidentiality of information				■
5. Simultaneous visualization of relevant information				■
6. Continuous monitoring	■		■	
7. Integration into existing models				
8. Purposeful tools and methodologies				

2.3.4 Role Model

The role model contains all roles defined in the process model. It is not limited to the roles directly involved in the project, but also includes organizational, legal, and other roles, thus representing all relevant stakeholders [39]. A role consists of at least a name, a description, and the assigned activities. Depending on the role and the project, a role can be assigned to exactly one person or to several people. Several people can take on the same role in parallel [46].

Since the role model and the concepts associated with the roles are strongly influenced by the project, the system, and the underlying process model, only a general role model will be developed in this section. This generic model can then be adapted and extended during the concrete implementation of the process.

The foundation of the role model is formed by the twelve roles of systems engineering described by Sheard [47]. These twelve roles are not described in detail, but are aggregated into the four roles shown in Table 8.

Table 8 – Aggregated roles

Aggregated role	Roles by Sheard[2]
Requirements Owner	Requirements Owner Customer Interface
System Designer	System Designer Glue among Subsystems Information Manager Process Engineer
System Support	System Analyst Validation and Verification Engineer Technical Manager Logistics and Operations Engineer
Coordinator	Coordinator

The role **Requirements Owner** (RO) combines both the customer's view of the system and traditional requirements engineering. From the perspective of the CS, the requirements owner acts as a requirements generator towards the SoS. As part of the SoS, on the other hand, the requirements owner places requirements on the participating systems. Furthermore, the role forms the interface to external sources of requirements and changing framework conditions.

The **System Designer** (SD) summarizes all technical issues. This includes the design of the systems as well as knowledge about interfaces, information, and data or related engineering processes. From the CS point of view, the role accepts and implements requirements in all these areas. As part of the SoS, it defines, for example, interfaces or standards that need to be implemented by the CS.

The role **System Support** (SS) is responsible for compliance with the requirements and for correct functionality from both the CS and the SoS point of view. In addition, the provision of supporting infrastructure is part of this role.

The **Coordinator** (CO) is responsible for coordination and consensus building. The coordinator's task consists primarily of mediation and conflict resolution between diverse stakeholders.

[2] In [47], Sheard also defines the role »Classified Ads. Systems Engineering", essentially denoting unspecified »other", problem-specific roles. In Table 8, we ignored this category.

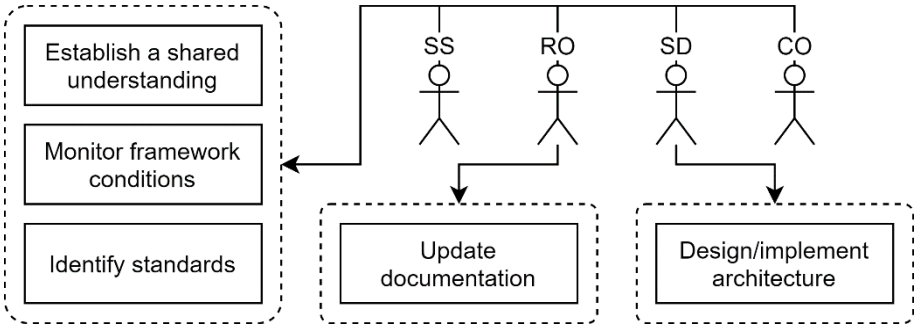


Figure 5 – Roles and their activities

The actual realization of the roles depends on many factors that are not further discussed here. An important factor is the form of the SoS. While in a directed SoS (see Section 2.1) the roles of the requirements owner or system designer of the SoS are usually implemented by a central authority, it becomes more likely with increasing independence that these roles will be implemented by a board consisting of the different requirements owners and system designers of the CS.

2.3.5 Sequence Model

The sequence model provides a chronological frame for the aforementioned elements. Phases and milestones are used for the organization and serve as container elements, which can be populated by components of the previously described submodels. Milestones can be used to define phase transitions as well as necessary inputs and outputs between the phases [48].

The exact integration of activities into a process model depends on the model itself. Nevertheless, the order of the activities and the iterative realization derived from *Requirement 6 (Enable continuous monitoring)* should be retained.

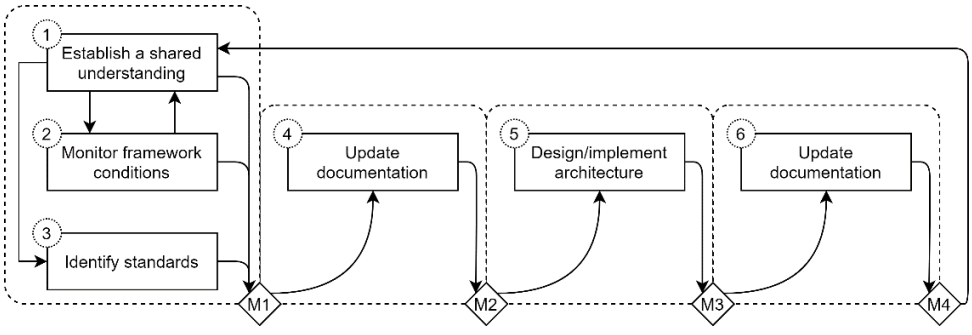


Figure 6 – Sequence of activities

The first step, as depicted in Figure 6, is to create the shared understanding. Next is the review of the surrounding conditions, with the results being passed back to the first step, which may trigger an adjustment. The third step is the identification of existing standards and standardization gaps. By completing Step 3, information gathering and knowledge generation are finished for this iteration. This state is recorded in milestone M1 and is coupled to the generated and externalized knowledge as artifacts “Shared Understanding”, “Ubiquitous Language”, and “Standards and Standardization Gaps”. The newly generated results are passed to the documentation in Step 4. The result of this step is a new version of the artifact Body of Knowledge. The completion of the update also marks the achievement of milestone M2. The documented results are used as input for the design and implementation of the SoS architecture. Step 5 is covered by external processes that are not part of the process described here. The completion of the architectural design based on the information collected in the previous milestones leads to the achievement of milestone M3. Both the architectural design and implementation decisions must be documented and affected parts of the documentation must be adapted. The updated documentation in milestone M4 contains all the information and artifacts of an iteration and serves as a starting point for the next iteration.

2.3.6 Combining the Sub-Models

Combining the four sub-models, we obtain the overall process shown in Figure 7. The process consists of six steps, which must be completed for each iteration.

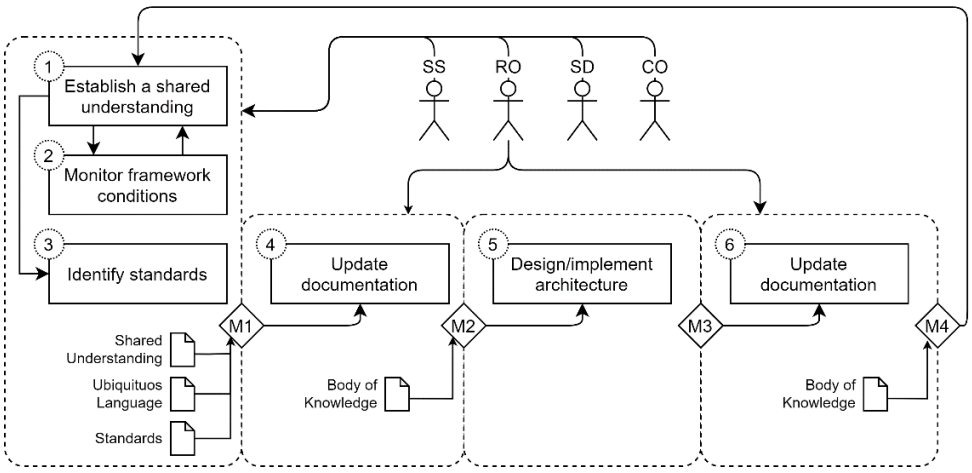


Figure 7 – Combining the sub-models into the overall process

The abstract nature of the artifacts, activities, and roles allows the process to be implemented by a wide variety of methods and tools, as long as they can meet the described characteristics of the components. Table 9 compares the process with the requirements defined at the beginning and shows which process sub-model

fulfills the respective requirement. The role model is part of the process but has no direct relation to the described requirements.

Table 9 – Assignment of process sub-models to their realized requirements

Requirement	Realized by
1. Establish a shared understanding	Artifact Model Activity Model
2. Enable collaboration	Artifact Model Activity Model
3. Interoperability	Artifact Model Activity Model
4. Confidentiality of information	Artifact Model Activity Model
5. Simultaneous visualization of relevant information	Artifact Model Activity Model
6. Continuous monitoring	Activity Model Sequence Model

Each requirement is fulfilled by parts from two sub-models. It is quite obvious that a requirement is fulfilled by an activity as well as by an artifact, since the artifact is the exact result of the activity. Since *Requirement 6 (Enable continuous monitoring)* is mainly addressed by the iterative procedure, it is also obvious that this requirement is fulfilled by parts of the activity model and by the sequence model.

2.4 Suggested Tools and Methods

After the process has been developed on a rather abstract level in Section 2.3, this section presents some concrete methods and tools that can be used to implement the process. The implementation presented in the following will center on the use of Lego Serious Play, reference architecture models, use cases according to IEC 62559, and a visualization tool developed especially for this purpose. The section ends with an outlined process for the problem formulation explained above.

2.4.1 Lego Serious Play

Lego® Serious Play® (LSP) is a facilitated approach that allows participants to work on complex topics using metaphorical Lego models and storytelling. Frick et al. [49] describe various application areas and tasks that can be worked on using LSP. For example, LSP can be used to

- find creative solutions to new types of problems,
- develop a shared idea,

- turn tacit into explicit knowledge,
- gain deeper insights and a better understanding of the goals and motivation of other participants,
- resolve tensions between different groups or persons.

These different application areas cover the activities for creating a shared understanding, including all related sub-activities. The wide range of applications, the use of different process components, and the experiences gained while using the methodology lead us to the conclusion that LSP is a suitable method for the realization of the process and thus meets *Requirement 8 (Use purposeful tools and methodologies).* In the following, some theoretical background will be presented. Then the principles and the steps of the method will be described, followed by the description of the method as part of the process.

2.4.1.1 Scientific Background

Scientifically, LSP is based on the theory of constructionism, on the concept of play and flow, and on the complex interaction between hand and mind. Kristiansen and Rasmussen [50] describe LSP as "...a way of thinking with objects and through our hands to unleash creative energies, modes of thought, and ways of seeing that most adults have forgotten they even possessed".

2.4.1.2 Principles and Methodology

LSP is based on several basic rules and principles that define the behavior and approach required to implement the method. These principles and rules should ensure the success of the methodology, both for the task at hand and for each individual participant. They can be summarized as follows [49] [50] [51]:

- The Lego model is the answer to the question asked; there are no wrong answers.
- Each participant builds a model and shares the story of their model with the other participants.
- Think with your hands and tell the story of the model.
- Each participant has the interpretive authority over their own model.
- The story and meaning of the model must be accepted.
- Listen with your ears and eyes. Questions are asked only about the model or the story, not about the person.

Every LSP workshop opens with so-called skill building. This initial step helps the participants to familiarize themselves with the LSP rules and the core process as well as to learn basic construction techniques. In addition, the first reservations about so-called toys are overcome. Regardless of the actual topic of the workshop, the core process consists of four steps:

1. The facilitator formulates the question and introduces it to the participants.

2. The participants build and develop their Lego models with the corresponding story.

3. The participants share their Lego models and stories.

4. The participants reflect on their understanding of the models.

Seven application techniques (ATs) can be used to address the questions and objectives of the workshop. Each of these ATs is based on the core process described above. The sequence of the ATs is by no means linear. The only mandatory AT is the development of the individual models and thus AT1. For AT6 and AT7, it is recommended performing all previous ATs as well. The ATs are [50]:

1. Building individual models and stories: AT1 aims to help each participant access new and invisible knowledge and externalize tacit knowledge. By sharing individual models, knowledge, and feelings, thoughts are conveyed to the other participants.

2. Building shared models and stories: The goal in AT2 is to consolidate the individual models into one shared model. The consolidation process creates a common model that corresponds to the shared understanding of the participants.

3. Creating a landscape: The landscape model developed in AT3 intends to analyze and categorize the individual models. Similarities, differences, and patterns should be identified without losing details or changing the meaning of the individual models.

4. Making connections: The mapping and identification of relations between the Lego models – more precisely, between the story or the meaning of the models – takes place in AT4. The relations are also made out of Lego parts and can themselves transport a meaning (e.g., a chain can have different semantics than a bridge).

5. Building a system: AT5 is an extension of AT4. Connecting several models to form a complex system leads to interactions and to unforeseen influences on affected models becoming visible.

6. Playing emergence and decision: The prerequisite for AT6 is the system developed in AT5. Based on different scenarios and decisions, it is investigated how the system and its parts react to unforeseen dynamic events and what kind of emergent behavior it might exhibit.

7. Extracting Simple Guiding Principles: The “Simple Guiding Principles” describe principles that are intended to support strategic decision-making directly and in real time. They arise from the knowledge acquired in the previous ATs, in particular by observing the system behavior in AT6.

2.4.1.3 LSP Application

Its versatility makes LSP a suitable methodology for SoS projects. The focus on building the Lego model as well as activating creativity and hidden knowledge helps to leave familiar paths and find and develop novel solutions. These novel and innovative solutions are essential for SoSE where approaches from traditional systems engineering may fail or fall short.

Adherence to the LSP principles and rules implies that each participant is involved equally and throughout the entire process. The sub-activities “Cross-Domain discourse” and “Fostering active participation”, which are part of the activity “Establishing a shared understanding” (see Section 2.3.3.1), are therefore covered by the LSP methodology.

Blair [51] organizes the seven ATs into three levels that build on each other and grow in complexity and scope, as shown in Figure 8. The levels shown are annotated with the corresponding process activities and results.

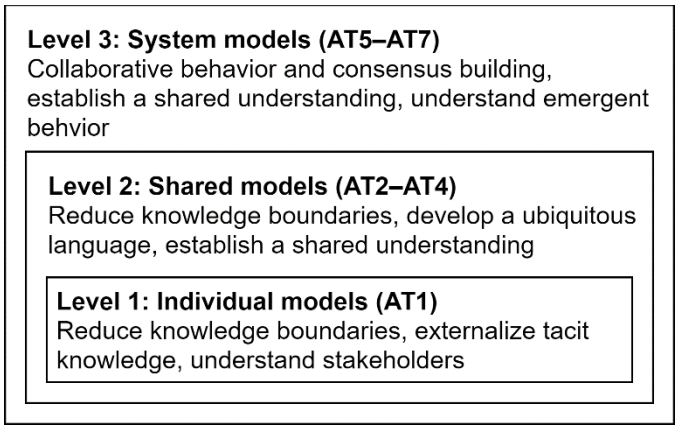


Figure 8 – AT layers and their related process components

The first level externalizes tacit knowledge, reveals the motivation and goals of the stakeholders, and provides insight into the mindset of other stakeholders. Linking the individual models in level 2 helps to better understand other stakeholders and strengthens the shared understanding. The necessary discussion and the identification of differences, similarities, and patterns support the development of the ubiquitous language. The third level also supports the activity “Establish a shared understanding”, but it goes even further. System models help simulate, understand, and evaluate the effects of different decisions and events on the system and its components. Thus, they support the planning and detection of possible emergent behavior as well as decision-making in democratic and consensus-based structures. Making these interrelationships visible and comprehensible contributes to the promotion of collaborative behavior and consensus-building mentioned in *Requirement 2 (Enable collaboration)*, which supports the process as a whole.

The models generated in the ATs can also be seen as the first form of documentation and are therefore part of the BoK (see Section 2.3.2.2). Since the collaborative models, in particular, represent all relevant information at the time of modeling from the point of view of the stakeholders, they also partially meet *Requirement 5 (Visualize all relevant information simultaneously)*, which is not a requirement for the process, but rather for the implementation of the BoK.

Due to the required physical presence of all participants in one place and the time and resources involved, it may be advantageous not to use the LSP methodology in every iteration. LSP is of particular importance when significant changes in the general conditions and objectives are to be expected or when crucial project milestones are reached. One of the most important stages in the process is the start of the project to ensure that all stakeholders share a common vision and can develop

a shared mental model. Table 10 summarizes the SoSE process components (see Section 2.3) realized by LSP.

Table 10 – Components of the SoSE process implemented by LSP

SoSE process components	Correspondence in LSP
Artifact model	
Shared Understanding	LSP Principles, AT1–AT6
Ubiquitous Language	LSP Principles, AT1–AT6
Standards and Standardization Gaps	-
Body of Knowledge	LSP Models as Part of BoK
Activity model	
Establish a shared understanding	LSP Principles, AT1–AT6
Identify standards	-
Monitor framework conditions	LSP Principles, AT1–AT4
Update documentation	-
Further requirements	
Simultaneous visualization of relevant information	LSP Principles, AT2–AT5
Purposeful tools and methodologies	LSP Principles, AT1–AT6

2.4.2 Reference Architecture Models

A Reference Architecture Model (RAM) – such as the Smart Grid Architecture Model (SGAM) or the Reference Architecture Model Industry 4.0 (RAMI 4.0) – can be used to make the complexity in SoS manageable. The SGAM, as one of the first models of its kind, originally aimed at revealing gaps in standardization and advancing standardization in the Smart Grid [52]. Since then, the actual utilization of the model has turned into a development and communication tool, because it offers a holistic and sufficiently abstract view on complex system landscapes in the Smart Grid [53].

Both the initial intention and the current use of the models illustrate the suitability of reference architecture models as viable tools and thus meet *Requirement 8 (Use purposeful tools and methodologies).* The successful application in various projects and contexts confirms their versatility. Applications include, for example, requirements engineering [52] [54], the design of SoS [55] [56], or sharing knowledge and related discussions [57] [58].

In the remainder of this section, various reference architecture models will be introduced briefly, with the SGAM as the initial model being described first. Finally, the use of these models in the developed process will be discussed.

2.4.2.1 Models

The SGAM as well as its derivatives are three-dimensional models consisting of vertical, aligned layers. Each layer corresponds to an interoperability layer and shows a specific perspective on the system under consideration. The horizontal axes that span these layers correspond to domain-specific concepts.

The SGAM interoperability layers are based on the interoperability stack of the GridWise Architecture Council (GWAC). The stack represents the organizational, semantic, and syntactic aspects of interoperability of systems in the Smart Grid in eight layers and thus precisely addresses the aspects described in *Problem 3 (Knowledge Boundaries).* According to [59], the layers are not restricted to the Smart Grid, but remain valid in other domains and in the SoS context in general. Figure 9 shows the reduction of the eight layers of the GWAC stack to the five layers of the SGAM [60].

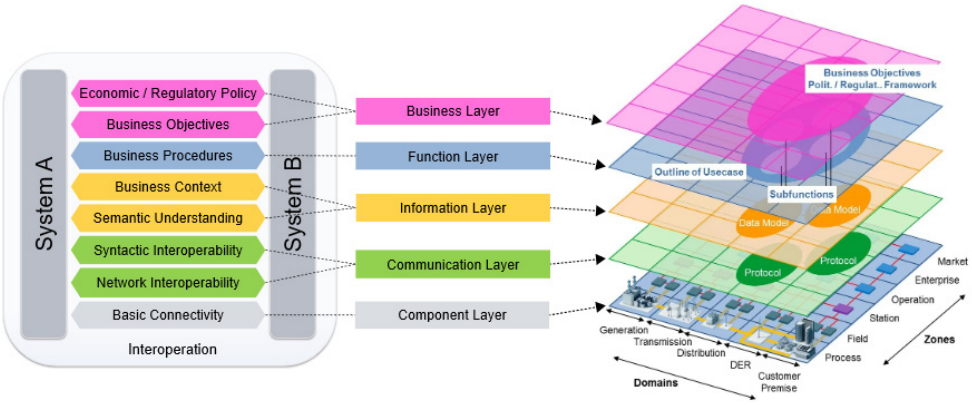


Figure 9 – Reduction of the eight-layered GWAC stack to the five SGAM layers [60]

According to [60], the Business Layer includes all aspects related to the business model and the business case. This encompasses not only the actual business processes but also regulatory and legal aspects. Located below it, the Function Layer represents all logical functions and services that are used to implement the business process. The information exchanged by the Service Layer components as well as their formats, standards, and modeling rules are mapped onto the Information Layer. The Communication Layer describes the logical communication infrastructure and its protocols, while the underlying Component Layer captures the components and actors in the Smart Grid. Each layer is spanned by the horizontal axes “domains” and “zones”, which on the one hand cover the energy conversion chain

in the Smart Grid and, on the other hand, consist of the automation pyramid, extended by the “Enterprise” and “Market” areas.

RAMI 4.0 represents the most advanced derivative of the SGAM. Its layers are, on one side, also spanned by the automation pyramid. However, the extension does not include “Enterprise” and “Market”, but, according to the idea of the interconnected Industry 4.0, “Enterprise” and “Connected World”. At the lower end of the automation pyramid, “Product” is also added, since it is of particular importance in the context of Industry 4.0. On the other side of each layer, the product lifecycle is depicted, which is important for product and system development. The interoperability levels of the SGAM have been adopted as far as possible. The Component Layer is split into the layers “Asset” and “Integration”. The asset layer corresponds to SGAM’s Component Layer and represents real assets in the physical world, while the Integration Layer represents the transition of physical assets into the digital world.

Besides SGAM and RAMI 4.0, several other models for different domains have been developed. These include, for example, the Smart City Infrastructure Architecture Model (SCIAM), the Maritime Architecture Framework (MAF), and the Legal Reference Architecture for Industry 4.0 (ju-RAMI 4.0), which are displayed in Figure 10.

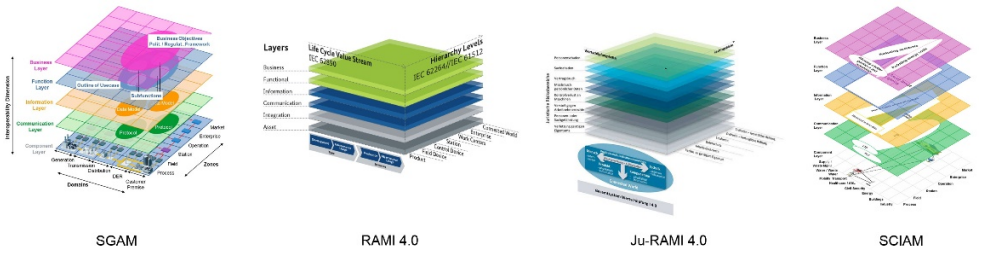


Figure 10 – Sketch of different RAMs for different domains

2.4.2.2 Model Application

The original purpose of standardization and the provision of a holistic view of the system enable the purposeful use of reference architecture models in many of our process components.

The combined representation of all relevant information within a single model on layers organized by context already fulfills *Requirement 5 (Visualize all relevant information simultaneously).* By separating these layers – according to the principle of Separation of Concerns – we can narrow our view on the information that is relevant for one specific task or question. In addition, the absence of formal requirements for filling out the respective layers allows almost any project-relevant

information to be located and captured within the model, making a reference architecture model a promising implementation of the artifact “Body of Knowledge”. By representing the logical system architecture within the RAM, the model can also serve as a development tool. By always reflecting the logical architecture in the RAM, this also corresponds to a continuous update of the documentation and thus partially fulfills the activities “Update Documentation” and “Monitor Framework Conditions”.

The terms used in the model and their meanings provide both a starting point and a framework within which the entire SoS with its ubiquitous language, a shared understanding, and the use of existing standards can evolve. For instance, RAMI 4.0 was developed with the idea of harmonizing the different user perspectives on the topic of Industry 4.0. The identification standards aresupported by the grid created on the interoperability layers. By locating the standards on the respective cells corresponding to the horizontal axes, they are directly visible for systems that logically belong to those cells.

Because RAMs can be used simultaneously as a tool for communication, documentation, and development, they should be used in almost every step of the process.

2.4.3 Use Cases

Use cases (UC) describe the behavior of a system from an actor's point of view when interacting with the system. Actors are not limited to human beings, but can represent roles, organizations, or other systems. Among the actors involved in a use case, there is a so-called Primary Actor who triggers the use case by their initial interaction and thus wants to achieve a certain business goal [53]. ISO/IEC 19505-2:2012 [61] defines a use case as “the specification of a set of actions performed by a system, which yields an observable result that is, typically, of value for one or more actors or other stakeholders of the system”.

Depending on the initial action, the system’s state, and other, possibly unknown, variables, a use case can take different paths with different results. Each of these paths is captured as a possible scenario, which itself consists of at least one step. Each step describes an action to be performed, which may also include interaction or communication with other actors [53].

A use case is usually based on a more abstract business case, which itself does not provide any technical details. The business objectives captured in the business case are further specified and realized by use cases of different granularities. Gottschalk et al. [53] differentiate between “high level”, “generic”, “specialized” and “individual” use cases. The level of abstraction decreases in this order, while the level of detail increases accordingly.

Using a high-level use case as a starting point for further specification corresponds to a top-down approach. Alternatively, a bottom-up approach can be followed. In this case, individual and specialized use cases are defined first to fulfill the business case. The more abstract use cases are then derived from these individual use cases. The bottom-up approach is mainly used when many different perspectives and points of view need to be combined.

2.4.3.1 IEC 62559 Use Case Methodology

A structured and standardized process for eliciting and managing UCs is described in the four parts of the IEC 62559 – Use case methodology series. Part 1 lays the foundation for collaborative elicitation and sharing of use cases by describing prerequisites and requirements for a so-called UC repository. The necessary exchange and serialization formats are described in Part 3 of the series. Part 4 concludes the series with best practices for UC elicitation.

Part 2 of the standard [62] provides a template for the structured and standardized elicitation of use cases. This standardized elicitation enables the comparison, aggregation, and abstraction of different UCs and thus also the provision of potential blueprints for future use cases. In addition, the template helps record all information from the various stakeholders. To do so, it consists of eight sections, whose detailed content can be found in Appendix A:

1. Description: Section 1 of the use case template provides a description of the use case, collecting all general information about the intended goals of the use case. Additional information may also be provided here, such as preliminary assumptions, prerequisites, or the location within the domain.

2. Diagrams: Section 2 contains the diagrams of the use case. Though UML diagrams are usually used, any type of drawing and representation is allowed. UML use case, activity, and sequence diagrams, in particular, provide a good understanding of the flow of the use case.

3. Technical details: The technical details, comprising lists of the actors and references, are covered in Section 3 of the template. The actors are listed with their names, types, descriptions, and optionally with additional information specific to the use case.

4. Step-by-step analysis: The step-by-step analysis of the use case in Section 4 of the template describes the possible scenarios of the use case. The scenarios should correspond to the diagrams in Section 2 of the template. Each scenario consists of at least one step describing an action or the interaction with other actors.

5. Information exchanged: Section 5 of the template presents the information exchanged in the scenario steps. An information object is specified by an identifier for referencing, a name, and a description. In addition, requirements (defined in template Section 6) can be added to the information object specifications.

6. Requirements: Section 6 of the template identifies the requirements that are relevant in the scope of the use case. They are grouped into categories and can be referenced, for instance, in Section 4 or Section 5.

7. Common terms and definitions: Section 7 of the template corresponds to a glossary. Each important term used in the use case must be followed by its definition.

8. Custom information: Optionally, custom information can be added in Section 8. The information is described as a key-value pair. In addition, it is specified to which section this information refers.

Completing an IEC 62559-compliant use case requires a considerable amount of work, but it is a good way to capture the knowledge released and generated in a structured form during SoSE, for example, by using LSP. Use case descriptions also conveniently support the documentation of subsystems or individual functions required to achieve the goals of the SoS.

2.4.3.2 Use Cases and SGAM

RAMs can visually illustrate use cases, which, among other things, can support visual analysis of use cases and simplify communication among engineers. Since the SGAM is the most sophisticated and well-known RAM for use case visualization, this section focuses on the relationship between IEC 62559-compliant use cases and the SGAM.

Different approaches can be used to develop SGAM models based on the IEC 62559 use case methodology. However, it is essential that as much information as possible about the use case is known at the start of modeling. Although the IEC 62559-2 use case template is not part of SGAM modeling, it contains all information necessary to realize a use case with SGAM. In the EU mandate M/490, a mapping process from use cases to SGAM models is described. The process can be carried out using one of three different approaches. Regardless of the approach selected, there are always six steps, with the first one being “Use Case Analysis”. Gottschalk et al. [53] describe the three possible approaches as follows:

- Mixed-up: The expert starts developing the component layer by mapping the actors involved and their relationships. The relationships of the actors

stem from the steps of the use case template. Here, the relationships between the actors represent communication paths (power flow, messages sent, etc.). Next, the Business Layer is defined. The business objects and rules required for this activity are listed in the use case description. Based on the information exchanged in the component, the Business Layer, the Function Layer, the Information Layer, and the Communication Layer are developed.

- Top-down: The top-down approach starts with the elaboration of the Business Layer, followed by a top-down progression through the layers of the SGAM. This approach is often used when a high-level or generic use case is described. The specific implementation of the use case is unknown or unimportant at this point.

- Bottom-up: The bottom-up approach progresses inversely to the top-down approach. It is suitable for the implementation of individual use cases that describe a concrete implementation to realize a business case.

For the modeling of the individual layers, all required information is contained in a properly completed use case template. When positioning the entities in the domains and zones, appropriate domain knowledge is required. Incomplete or empty layers indicate that the use case is not fully specified. However, it must be considered that a use case can be recorded in varying degrees of detail within the IEC 62559-2 use case template, and that the template leaves room for interpretation in this respect.

2.4.3.3 Use Case Application

In SoSE, use cases can provide a foundation for bringing together different perspectives of the stakeholders as well as serving as a communication platform for productive cooperation. They also promote communication among subsystems in subsequent analyses of the system, such as risk or security analyses.

Together with RAMs, the use case methodology also facilitates the structured documentation of business process flows in the development of corresponding architectures with their static components and data models. In addition, IEC 62559 provides a template for capturing the possibly still unstructured technical knowledge acquired through LSP in a structured way and for using it as a starting point for further elaboration.

However, this type of modeling focuses primarily on functionalities, while non-functional requirements might not be adequately captured in UC models. Yet, it is

precisely these non-functional requirements that are important for both the constituent systems and the success of the entire SoS. We will return to this issue in Section 3.

2.4.4 Visualization

As described in the previous sections, RAMs are an efficient and effective way to implement the artifact “Body of Knowledge”. In fact, we recommend using RAMs in as many process steps as possible. In order to meet these two requirements, we need to implement RAMs in such a way that they can be used interactively, from distributed locations, and persistently. The realization of RAMs in the form of an interactive visualization represents such an implementation.

In the following, the added value of visualization will be explained. Then a prototypical, local implementation will be presented as well as an outlook on a distributed and collaborative solution. Finally, assuming a collaborative solution, the application of visualization within the SoSE process will be examined.

2.4.4.1 Added Value

In [63], Munzner discusses the advantages of visual presentation of information over other forms of information transfer. Unlike sound, for example, the use of visual support is particularly well suited because several layers can be viewed and displayed simultaneously. In addition, the visual system is connected to the brain with high bandwidth and an essential part of the information processing takes place in parallel in the subconscious mind. In contrast, the transmission of many pieces of information at the same time through sound is not feasible or very limited because the information is presented and processed sequentially. Visualization therefore helps expand human abilities by being used as a form of external memory. The ability to spatially arrange the information within this external memory supports both the search for information and the recognition of complex relationships.

The visualization of complex subjects has long been used in many different domains to understand, analyze, and evaluate issues both better and faster. Software is also visualized on different levels of abstraction [64]. The textual representation of the source code and supporting tools such as code highlighting, debugger, and profiler can be considered the least abstract level of visualization, while the visualization of the architecture is done on a much more abstract level. The goal of visualization on the level of software architecture – or, more generally, on that of system architecture – is to make the complex connections and design decisions comprehensible and communicable as quickly and clearly as possible [65].

The presentation of all information in a uniform context also takes up the principle of a ubiquitous language for visual representation. In addition, a uniform presentation across all perspectives avoids constant changes between several views and thus reduces the cognitive load. The representation of the internal structure and the details of the CS by the same view in which the entire SoS is represented also helps to further reduce the cognitive load of the stakeholders and prevents disruption in the language and description used to document the architecture.

The structured organization of all information in an interactive visualization corresponds to the capability to capture and display the entire project knowledge base. Thus, the visualization becomes a central artifact in which all further partial artifacts are merged. The ability to interact with the model, to view and to modify it, creates a tool that can be used in all phases and different process steps of the project lifecycle.

2.4.4.2 Prototype

Our prototype supports the local visualization and manipulation of system architectures based on established RAMs. The prototype represents reference architecture models with the system architecture modeled there in three-dimensional space. This three-dimensional representation is the most important component on the user interface, therefore it is given the most space. According to the so-called Center Stage Pattern, it is located in the center of the application, while additional information and control elements are arranged around the central area.

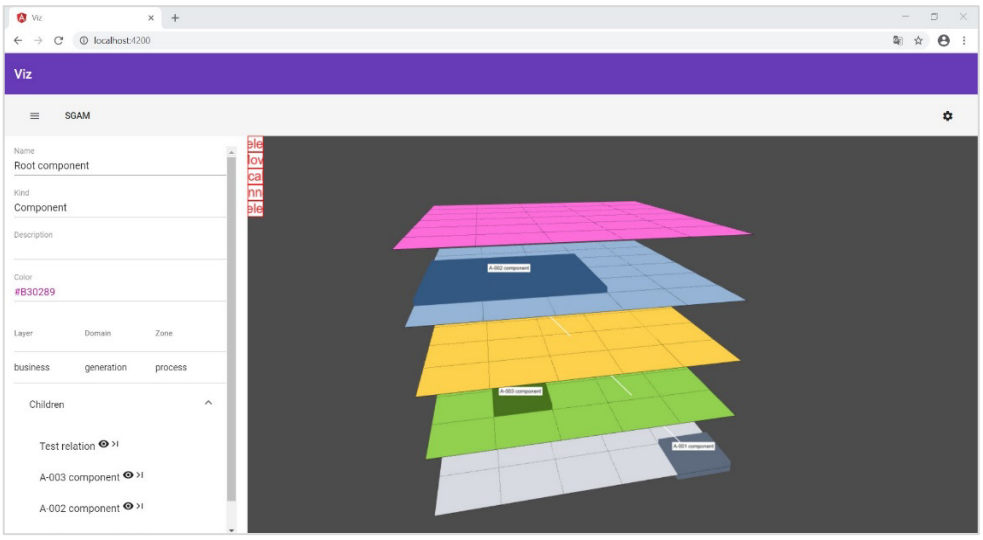


Figure 11 – Prototype

The mouse and the arrow keys can be used to change the view and to navigate through the model. The buttons in the upper left corner allow, for example, the addition, deletion, or modification of components, which are displayed in the form of cubes. Double-clicking on a component loads the internal structure of this component, which is also displayed in the corresponding RAM. These nested models allow, among other things, the encapsulation of information that is not important for the next-higher level of abstraction. This improves the clarity of complex system architecture models.

In order to meet the requirements and achieve the desired usage in all process and project steps, the prototype still needs to be extended by functionality for distributed and collaborative work. To provide a consistent state between all participants for several simultaneous users, employing a client-server architecture is recommended.

2.4.4.3 Utilization

The visualization prototype is a tool that can be used in a domain-oriented manner and (in a future version) collaboratively by all stakeholders, regardless of their location. Thus, the application fulfills *Requirement 8 (Use purposeful tools and methodologies).* Provided that concepts for access control are implemented in the final version, the application also meets *Requirement 4 (Ensure confidentiality of information).* To ensure that the application is actually used in the project, we also need to pay attention to the realization of a positive user experience. The option to use the tool online is also important for all process components and requirements.

The interactive representation of all information as well as the various interaction options help create a shared understanding and recall the understanding already represented in the model. Relationships between the individual components can be shown directly through connections and, for example, by highlighting all components that would be affected by a modification. Capturing all information, from anecdotal information to relevant standards, makes the visualization a powerful way to reflect the complete BoK. Continuous use as a communication and development tool by all stakeholders ensures that both the review of all requirements and the documentation can be done with minimal additional effort. The RAM and the vocabulary used in the model correspond to the project's own ubiquitous language.

2.4.5 Outline of a Possible Process

Together, the four tools and methodologies presented in Section 2.4 can implement the different process components with their associated requirements. Consequently, the combination of these three independent components enables the

realization of the abstract process across the entire project lifecycle. Figure 12 assigns the tools to the respective process steps and illustrates their influence on the given step. The artifacts that can be mapped by tools are limited to those that can be captured by the visualization of the RAM. Although the ubiquitous language and the shared understanding are reflected in the use cases and the BoK, they primarily emerge in the minds of the stakeholders.

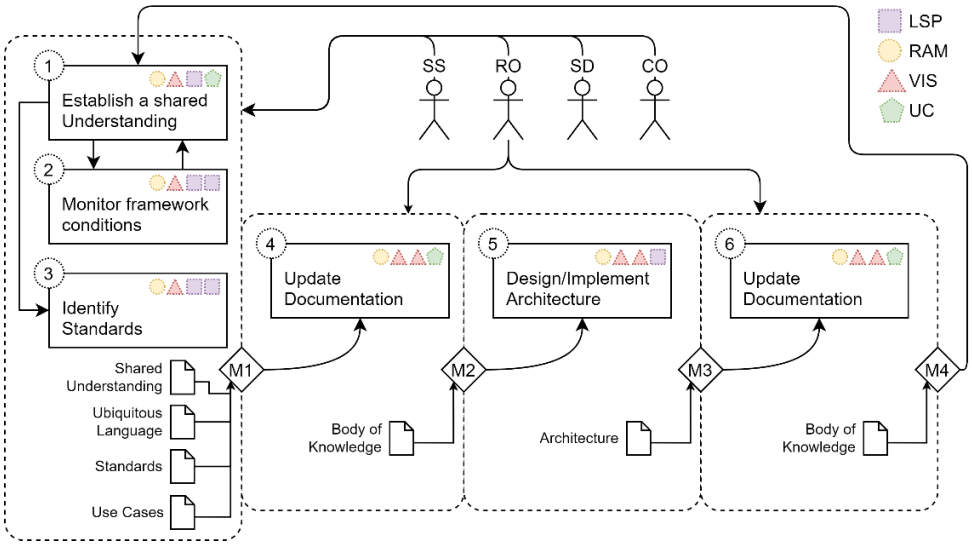


Figure 12 – Assignment of tools to process steps

LSP plays an important role in the initial phase and during extensive changes to maintain the shared understanding or to transform it as the project progresses. In addition, regularly scheduled LSP workshops promote cooperative behavior and enable the anticipation of potential emergent behavior. However, due to the physical presence required, LSP is not suitable for close collaboration in groups of highly geographically dispersed stakeholders on a daily basis.

For effective usage, employing a domain-oriented RAM such as SGAM or RAMI 4.0 is recommended. By mapping relevant concepts and areas as well as the corresponding interoperability levels, RAMs can be used for planning as well as for the implementation, evolution, and documentation of an SoS. In order to benefit from all advantages of a RAM, a distributed, interactive, and three-dimensional implementation of the RAM in a supporting tool is necessary. The resulting collaboration capabilities allow even widely distributed stakeholders to work together on a ubiquitous language, a shared understanding, and ultimately on engineering solutions.

3 Quality Requirements Analysis

To devise a suitable system design, we need to know the requirements of the various stakeholders. Often, stakeholders tend to emphasize functional requirements, which can be nicely specified as use cases. However, during requirements analysis, non-functional aspects such as performance, scalability, maintainability, or user-friendliness must also be considered carefully. For critical systems – and the Smart Grid is probably one of the most critical infrastructures – dependability, which includes properties such as safety, security, and resilience, is a very important requirement. In addition, systems that handle person-identifiable data need to fulfill privacy requirements.

In this chapter, we look into techniques for requirements elicitation and analysis, emphasizing non-functional system qualities and their assurance.

3.1 Requirements Analysis based on Use Case Modeling

Use case modeling is a centerpiece of the engineering best practices recommended by IEC Systems Committee Smart Energy standards. They provide basic information to advance the engineering process. This information can be refined into more sophisticated models as the use case specifications are enriched with more information from the stakeholders.

The IEC 62559-2 use case template has proven its value in practice, but it has some limitations regarding non-functional quality requirements. One problem is that its main focus is on functional views, that is, the process flow, the information exchanged, and the intended functions. However, the aspect of unintended behavior is neglected. Undesirable behavior can originate from an adversary triggering a system function that was not intended.

The concept of a so-called misuse case is one possible extension of the IEC 62559 template, but ideally, use cases should treat safety and security constraints or privacy needs as core aspects that are firmly linked to the individual parts of the use case description. For instance, the use case should state pre- and post-conditions as obligations for critical interactions between user and system, and it should assign quality attributes to input or output data of sensitive processing steps.

Moreover, a use case description should not only indicate the quality requirements. It should also provide some reasoning that – based on the use case specification with the stated constraints – the system will actually provide its functional-

ity with the required quality. Such a rationale is particularly important, if the system under development is a critical infrastructure for which we need assurance that the dependability and privacy requirements are definitely met.

3.2 Modeling Critical Requirements with Assurance Cases

Some qualities are mission-critical for the system under development. When valuable system assets, the well-being of system users, or the integrity of a system's environment are at stake, we need to ensure that the system reliably and correctly fulfills its intended purpose, especially its safety and security goals.

In these cases, it is not enough to strive for the required qualities; the prospective users also need strong assurance that the system does, in fact, meet its design objectives with the claimed quality. Assurance cases provide a formal way to reason about critical system properties.

3.2.1 Introduction to Assurance Cases

An *assurance case* is a generalization of a safety case, the latter being a "clear, comprehensive and defensible argument that a system is acceptably safe to operate in a particular context" [66].

An alternative interpretation of an assurance case can be to consider it as the set of documents that collectively form the above argument. For the purposes of this document, we will be adopting the first definition, which focuses on the essence, that is, the argument, rather than the form, that is, the documentation.

Whereas a safety case aims at assuring the specific quality of safety, an assurance case can encompass virtually any system property of interest. This can include arguing about the degree of satisfaction of a specific quality property or about a collection of properties that are deemed important to the system stakeholders.

An assurance case typically establishes a set of high-level claims to be proven by its underlying argument. To this end, these claims can be decomposed into more detailed sub-claims that support the former. This decomposition continues until the level of detail is appropriate for supporting the claims directly with evidence. The associated evidence should satisfy the base claims and, in doing so, propagate this satisfaction up the chain of sub-claims to the initial high-level claims. The argument is effectively the logical structure that forms the chain(s) from the evidence up to the high-level claims. Figure 13 provides an overview of this structure.

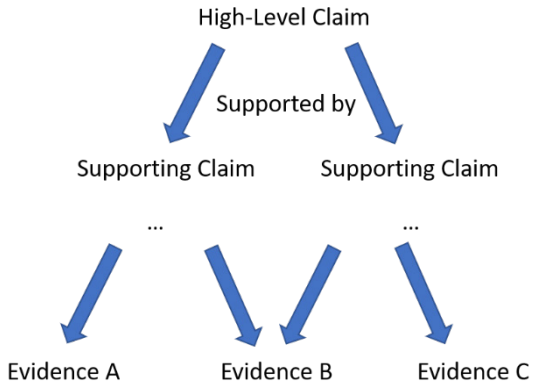


Figure 13 – Abstract Overview of Assurance Case Structure

3.2.2 Assurance Case Notations

In initial safety case practice, which persists even today, safety cases were composed using natural-language plain text [66]. While this approach is minimalistic and requires no special training or tools, it suffers from several weaknesses. Most notably, ambiguity and other communication errors can easily be introduced in the textual descriptions, and managing the complexity of the argument well enough so the stakeholders can comprehend it can be difficult. Moreover, as an assurance case often incorporates a plethora of evidence cited externally, the consumers of the assurance case are forced to repeatedly disrupt their flow of reading. Due to the large number of the system components involved, an assurance case can often grow to proportions that become difficult to manage using unstructured text. In particular, the structure of an assurance case itself can quickly become difficult to follow, leaving the reader uncertain about the validity of the argument. Even worse, changes in the argumentation, the evidence, or the intended claims may require disproportionate effort to correct, review, and verify the argument.

While such issues may not present grave problems when considering small-scale systems, they certainly become major barriers for the construction and comprehension of the assurance case for a moderately large system. Therefore, alternative approaches in the form of graphical notation systems have been introduced to supplement plain text and address these shortcomings.

Two popular notation systems are being employed today: the Goal Structuring Notation (GSN) [67] and the Claims-Arguments-Evidence (CAE) framework [68]. A recent development is the Structured Assurance Case Metamodel (SACM) [69], which is effectively a superset of GSN and introduces its own graphical notation. However, SACM is relatively new (Version 1 was published by the Object Management Group in 2013) and its graphical notation is even younger, having just been

published in 2020. For this reason, SACM will not be discussed further in this introduction.

GSN and CAE share many similarities in their philosophical origins, their modeling ontologies, and their visual notation approach. Without intending to disparage CAE in any way, we will adopt the GSN approach for the remainder of this document. Readers preferring the CAE approach should find it straightforward to adapt the GSN content in our report accordingly.

3.2.3 Overview of the Goal Structuring Notation (GSN)

GSN frames assurance as a problem needing to be solved by establishing *Goals*, then identifying appropriate intermediate supporting goals and eventually *Solutions* that can solve them. *Strategies* are used to link goals and intermediate goals and make their relationship – i.e., the argument linking them – explicit. Figure 14 shows a small example of a typical GSN argument.

In the example, G1 is the top-level goal, which claims that the system is acceptably safe to operate in a given context. The context of the system and its operating environment are linked via two corresponding *Context* elements, C1 and C2. C3 is another context element providing the relevant safety standard's definition of what is considered "acceptably safe". Context elements as well as other contextual elements (see Section 3.2.4) are typically placed horizontally to their associated element (or approximately horizontally if there are space limitations).

Strategy S1 supports G1 by explaining how G1 will be further supported; S1 effectively explains how support for G1 is broken down into sub-goals G2 and G3, each addressing individual hazards. The *Justification* for proceeding down this line of argument is that if all hazards are addressed in this way, the system is considered free from risk.

G2 and G3, in turn, claim that specific hazards for the system have been addressed; G4 aims at achieving completeness of the identified set of hazards by claiming that apart from those hazards listed in the model, there are no other hazards introducing additional risk to the system's operation.

Finally, solutions S1, S2, S3, and S4 underpin the final goals by providing evidence supporting each claim. For G2 and G3, their solutions consist of development artifacts (e.g., a documented Fault Tree Analysis) indicating that the risk of these hazards has been either eliminated or found to be acceptably low. For G4, evidence of the rigorous and systematic analysis carried out to identify the hazards and the high experience level of the analysts support the completeness claim.

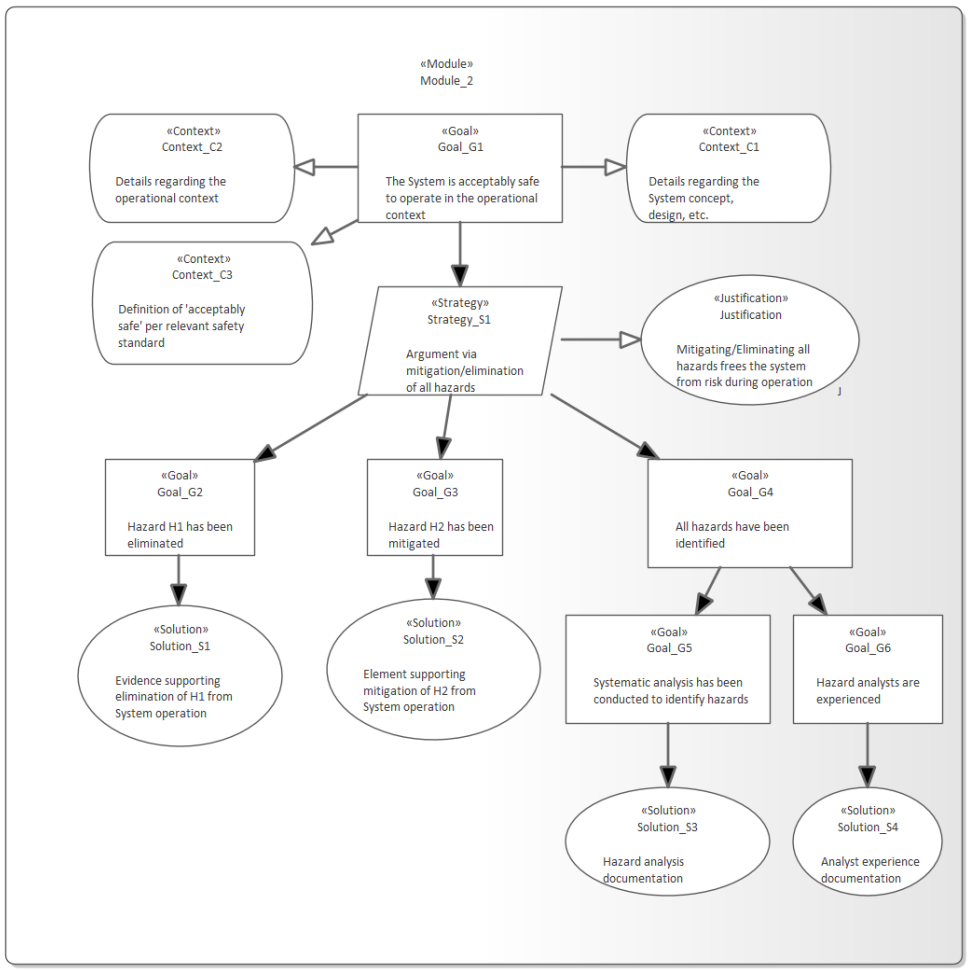


Figure 14 – A small GSN example

We should note that this example is very simplistic. It aims to illustrate the concepts of GSN modeling rather than provide a realistic assurance case.

3.2.4 GSN Basic Elements

The GSN Community Standard (v.2) [67] defines various elements for the construction of assurance cases and corresponding graphical notations, as shown in Table 11.

Table 11 – Basic GSN Elements

Element	Description	Graphical Notation
Goal	Used to establish a claim	«Goal» (Goal Identifier) <Goal Statement>
Strategy	Describes how a goal and its supporting goals relate to each other	«Strategy» (Strategy Identifier) <Strategy Statement>
Solution	Used to reference a piece of evidence	«Solution» (Solution Identifier) <Solution Statement>
Context	Contextual information	«Context» (Context Identifier) <Context Statement>
Justification	Expresses justification for the line of reasoning	«Justification» (Justification Identifier) <Justification Statement> J
Assumption	Expresses an assumption, i.e., a claim that will not be further developed as part of the argument	«Assumption» (Assumption Identifier) <Assumption Statement> A

The fundamental elements listed in Table 11 are usually enough to construct basic GSN arguments. However, to manage large-scale arguments, it is often useful to employ the module extension of GSN (see Subsection 3.2.6 below).

There is an additional decorator symbol, which denotes that the Goal or Strategy it decorates needs further development to be considered complete. Figure 15 shows how the decorator can be applied.

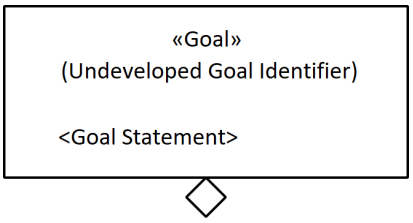


Figure 15 – "Undeveloped elements" notation

3.2.5 Linking Elements

Goals, Strategies, and Solutions are linked via "SupportedBy" relationships. Contextual elements (i.e., Contexts, Assumptions, or Justifications) are linked to the former via "InContextOf" relationships. These links are shown in Table 12.

Table 12 – GSN element links

Element	Graphical Notation
SupportedBy	
InContextOf	

3.2.6 GSN Modules

GSN *Modules* allow separating arguments into parts, each of which can designate "public" elements that can be cross-referenced by other GSN Modules. Figure 16 shows an example of this feature.

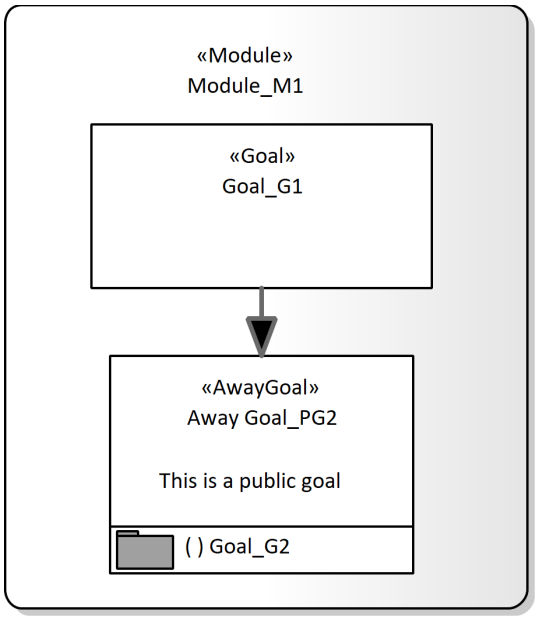


Figure 16 – Example Module M1 referencing another module

In the example, Module M1 contains Goal G1, which is supported by G2. However, G2 is a reference to a goal from a different module. Module M2, which actually contains G2, is shown in Figure 17. Note the small folder icon denoting the "public" status of G2.

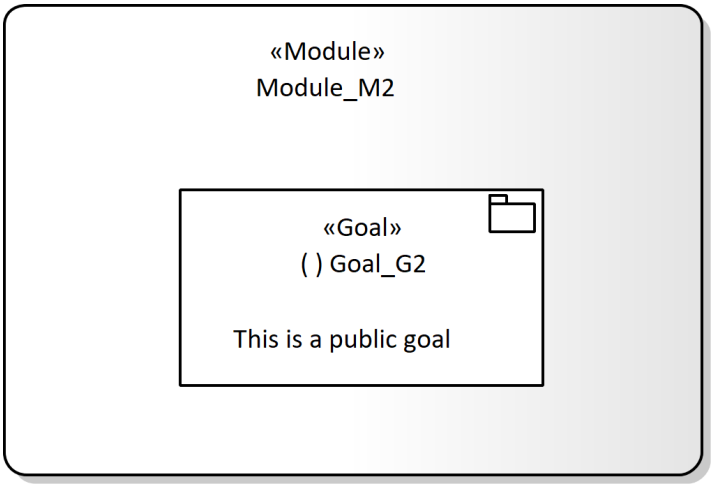


Figure 17 – Example Module M2 providing a public Goal for cross-referencing

Goal, Context, and Solution elements can all be annotated as "public" and referenced elsewhere. More advanced concepts regarding GSN Modules can be found in Annex B1 of the GSN Community Standard v.2 [67].

4 Eliciting Quality Requirements

The safety engineering domain has established methods and processes for systematically analyzing systems and system properties. As deviations from the intended purpose of a safety-critical system could harm people or the environment, proper engineering striving for completeness is obviously required. By adopting sophisticated safety methodologies, other important quality properties could also benefit from the considerable research effort spent on these approaches.

In this chapter, we will describe the main elements of our proposed modeling approach.

4.1 Quality Properties According to ISO/IEC 25010

During system development, quality models help to ensure that all relevant quality properties are adequately considered and that the needs of the stakeholders are addressed and reflected in the design. In the domain of software engineering, ISO/IEC 25010 [4] by the technical committee ISO/IEC JTC 1/SC 7 Software and systems engineering is currently the most comprehensive and most generally accepted quality standard. This norm, which replaced its predecessor ISO 9126, defines a quality model for the internal and external quality of the software product ("Product Quality Model"); it also provides model elements describing the applicability of a software system in the relevant application contexts from the viewpoint of the system users ("Quality in Use Model").

The ISO/IEC 25010 standard can serve as a checklist during use case modeling and design specification or – in subsequent lifecycle phases – as a testing guideline for system validation. The quality model provided by this standard helps determine which quality characteristics need to be considered during system development. If any of the listed qualities is critical for the system, this may warrant the elaboration of an assurance case as documented evidence that the system does, in fact, provide the required quality.

In the context of our modeling approach for enera, we strongly recommend the use of a quality model such as ISO 25010 or similar as a baseline for requirements analysis. A quality model provides additional evidence for arguing the completeness of an assurance case.

4.1.1 Product Quality Model

The ISO/IEC 25010 standard lists eight main categories of product quality attributes:

- Functional Suitability,
- Performance/Efficiency,
- Compatibility,
- Usability,
- Reliability,
- Security,
- Maintainability, and
- Portability.

Each of these categories comprises several sub-characteristics. Safety and security qualities – probably those qualities that most likely require assurance case modeling – are mainly reflected as sub-characteristics of *Reliability*, in particular

- Availability,
- Fault Tolerance,
- and Recoverability,

as well as these sub-characteristics of *Security*:

- Confidentiality,
- Integrity,
- Non-repudiation,
- Accountability, and
- Authenticity.

Some product characteristics that are generally considered important for safety and security also appear in the *Functional Suitability* category, in particular

- Functional correctness.

Interestingly, some relevant aspects of safety and security are missing in the list of product qualities; instead, the standard assigns them to the Quality-in-Use categories.

4.1.2 Quality-in-Use Model

Regarding the needs of the users who deploy and apply the system, ISO/IEC 25010 offers five main categories:

- Effectiveness,
- Efficiency,
- Satisfaction,
- Freedom from Risk, and
- Context Coverage.

These categories include several sub-characteristics that are often essential properties requiring a high level of assurance, particularly in safety- or security-critical application contexts, in particular:

- Trust (a sub-characteristic of *Satisfaction*):
  Degree to which a user or other stakeholder has confidence that the system will behave as intended.

- Health and Safety Risk Mitigation (a sub-characteristic of *Freedom from Risk*):
  Degree to which the system mitigates the potential risk to people in the intended contexts of use.

- Environmental Risk Mitigation (also a sub-characteristic of *Freedom from Risk*):
  Degree to which the system mitigates the potential risk to property or the environment in the intended contexts of use.

Especially the latter two sub-characteristics are key properties in terms of system safety and are most likely to warrant closer inspection using assurance case modeling.

4.1.3 Qualities Not Considered in ISO/IEC 25010

Remarkably, ISO/IEC 25010 does not explicitly address data privacy – that is, characteristics referring to the protection of person-identifiable information – and corresponding characteristics to ensure informational self-determination and transparency regarding the processing of personal data. The only characteristic vaguely referring to these types of qualities is confidentiality. This lack of privacy awareness may be due to the early publication date of the standard: It appeared long before the European General Data Protection Regulation (GDPR) put more emphasis on this quality aspect.

Some of the apparent omissions of ISO/IEC 25010 regarding data privacy are addressed in a related quality standard for data quality, ISO/IEC 25012 [70], such as

- Accuracy,
- Completeness,

- Currentness, or
- Accessibility.

Other privacy characteristics, such as data parsimony, need to be derived directly from the corresponding privacy regulations if the processing of personal data is intended. As a result, our method must resort to additional domain- or quality-specific standards if the system under development requires specific critical qualities not mentioned in the generic quality models such as ISO/IEC 25010.

4.2 Use Case Enhancements

The IEC use case template has gained significant momentum in the community. However, the use case methodology according to IEC 62559 offers only limited support for working out safety- or security-specific requirements and documenting their interrelationships. Requirements relating to functional safety or information security are just appended as “Safety Considerations” or “Security Considerations” to the description of the transaction steps. In addition, the requirements engineer may explicitly specify security objectives as “Objectives” of a use case; they may also be noted in the form of a mandatory post-condition of an application scenario. Apart from this, the use case template offers little to no specific guidance on these types of quality requirements.[3]

Basically, the use case methodology according to IEC 62559 treats security or safety aspects in the same way as any other non-functional requirement, such as performance, usability, maintainability, or scalability. The pre-standard IEC/PAS 62559 [71] merely recommended that potential security risks should always be examined when analyzing the use cases and that it should be checked which system assets are at risk and what kinds of vulnerabilities they may exhibit.

However, the application description for the use case template according to IEC 62559 does not specify how, for example, safety requirements or necessary preconditions can be systematically elicited or examined for consistency and completeness. Supplementary procedures and models are required for these tasks – which ones to use is left to users with domain knowledge.

For even more comprehensive support for security and safety analyses, additional information beyond the attributes provided in the template would be desirable, such as:

[3] In enera, a first suggestion for integrating security into the IEC 62559 use case template to support an information security management system (ISMS) was proposed and documented in the final report of the project [72, pp. 130-140].

- A listing of the safety assumptions: preconditions for a safe execution of the use case;
- A listing of foreseeable misuse cases together with the underlying motives for misuse;
- A survey of possible attackers (threat agents) and their characteristic attributes, such as available time and resources, skill level, or motives for attack;
- Anti-goals to be excluded: Events that must not happen under any circumstances;
- Functional assets (i.e., processes and interfaces) involved and their protection requirements;
- Physical assets (i.e., hardware) involved and their protection requirements.

However, given the scope of the existing use case template and the difficulties of the stakeholders in providing meaningful information on the required attributes, one has to consider very carefully which of the information that is desirable in principle should actually find its way into the use case template.

Especially for systems like the Smart Grid that are subject to constant change and emerging new cyber-threats, continuous security management is a challenge. Unfortunately, the use case primarily documents the *results of* a security requirement assessment: How these findings came about is not always so readily comprehensible.

In order to maintain an overview of whether all relevant safety goals of a Smart Grid design are still achieved – despite constant changes and expansions of the energy network and despite evolving security threats – , the reasoning as to why the system may be considered safe must be comprehensible and verifiable at any time. Here, Assurance Cases, as introduced in Section 3.2, offer a way to relate safety or security goals, assumptions, reasoning strategies, justifications, safeguards, security countermeasures, and corresponding evidence to each other. They promote understanding of where the individual safety and security requirements originate and how they interact to ensure that a higher-level safety objective is actually achieved. If the system changes, the argumentation chain modeled in GSN can be run through again to check whether all assumptions and conclusions are still valid and whether the solution modules provided for solving the safety problem together still meet the specified goal. If system changes cause a gap in the

underlying safety strategy, this shortcoming becomes more easily apparent in the GSN model than in a result-oriented requirements or use case specification.

In the enera project, we carried out a case study on an electric vehicle charging use case to illustrate how basic functional availability goals, safety goals, or IT security goals can be refined step by step into hierarchically structured sub-goals [72, pp. 180-186]. Using GSN, these sub-goal models are iteratively reduced until we finally obtain immediately evident solution modules. Along this refinement hierarchy, both the argumentation strategy and its justification as well as the assumptions made in the process can be recorded in corresponding annotations. Moreover, the specific context and its restrictions in which this chain of arguments is valid can be documented. In the event of system changes, these arguments are thus accessible for review.

Table 13 shows how the use case model can be enriched by means of GSN to justify a protection requirement more comprehensibly and to make it more maintainable (see Appendix A for reference to the respective UC template sections). However, relevant smart grid standards do not currently consider GSN as a modeling approach for use case specifications.

Table 13 – Assignment matrix GSN to UCMR model elements

GSN Element	Use Case Element	Recommendations for IEC 62559 UC Template
Goal	UC Section 1.3: Objective UC Section 6: Requirement	Integration of objective ID for easy cross-referencing.
Strategy	–	A conclusive connection between a goal and its sub-goals or requirements should be documented.
Solution	UC Section 8: Custom information	A solution refers to evidence that a goal or requirement is fulfilled. The UC template does not list this element explicitly yet, but this could be realized by a key-value pair in Section 8 of the template.
Context	UC Section 1.3: Scope	Documented only for the entire use case and not yet specific to sub-goals or strategies.
Assumption	UC Section 1.6: Assumptions	Assumptions can already be documented, but without reference to other UC elements.
Justification	–	Integrate the capability to explicitly document a rationale for a goal or strategy.

GSN Element	Use Case Element	Recommendations for IEC 62559 UC Template
SupportedBy	Reference to element with ID	Currently, only references to elements with an ID are feasible. Provide a continuous chain from goal to requirement.
InContextOf	–	Provide references to context, assumptions, and rationale.

The remainder of this section will focus on our proposed extensions and recommendations, motivate their need, and document the current status of the work.

4.3 Assurance Case as a Guide for Properly Addressing Quality Properties

When considering all quality properties that are important for the stakeholders of a system, a systematic guideline assures the coverage of the quality properties already during requirements elicitation. As each quality property can entail several concrete quality goals for the system, a comprehensible and traceable elicitation process becomes more important as the complexity of the system increases.

Such a systematic approach is provided by an assurance case, which includes process steps as well as resulting artifacts. In Figure 18, we derived the top goal shown with its quality properties, explained in Section 4.1, as a suitable starting point.

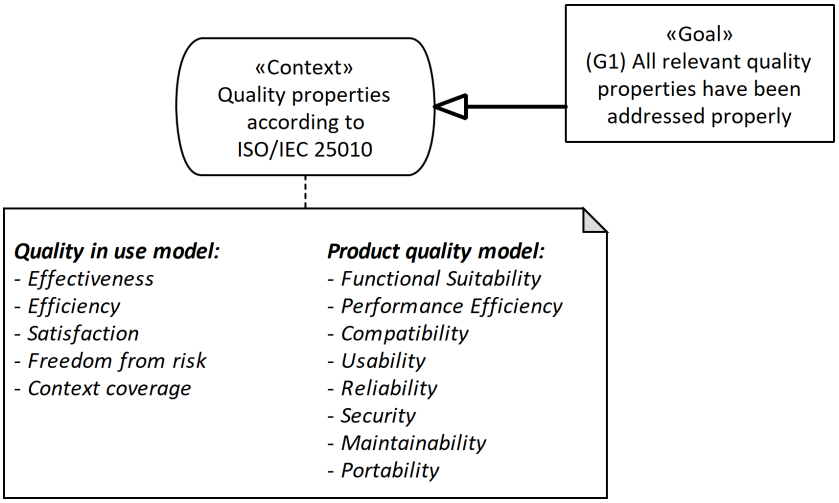


Figure 18 – Top goal of the assurance case for quality requirements

Goal G1 serves as the top goal of quality engineering and is formulated as

(G1) All relevant quality properties have been addressed properly.

The quality properties mentioned in G1 are further described by a context element, which restricts the goal to properties of the ISO/IEC 25010 only. This coarse-grained goal G1 must be divided into more fine-grained sub-goals in order to prove its fulfillment by corresponding evidence artifacts.

4.3.1 Refinement of the Top Goal

We follow the strategy of first identifying the relevant quality properties. Not all quality properties may be relevant to the stakeholders, so excluding some at the beginning may be useful. Relevant quality properties should lead to a corresponding sub-goal that claims their proper consideration. This process is expressed in the Activity Diagram shown in Figure 19.

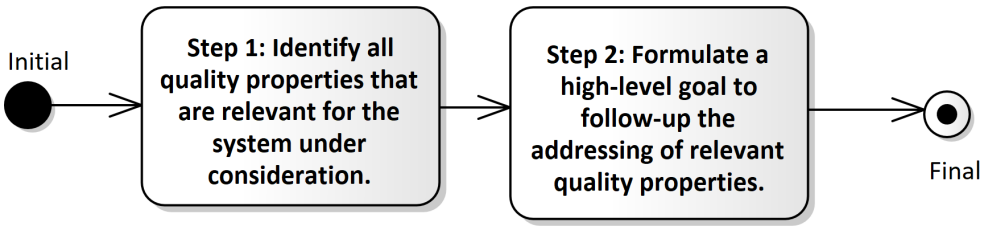


Figure 19 – Supporting activities to properly address relevant quality properties

To guide the engineer through these and further process steps, we have developed corresponding patterns for the assurance case that realize the processes. The process in Figure 19 leads to the assurance case pattern shown in Figure 20.

Step 1, which addresses the identification of the relevant quality properties, is transformed into Goal G2, formulated as

(G2) All relevant quality properties have been identified

which is further developed in a separate GSN module for the sake of clarity in Section 4.3.2 (for an introduction to GSN modules, see Section 3.2.6).

In Step 2 of the process in Figure 19, a sub-goal is formulated to follow-up the addressing of a relevant quality property. Therefore, we developed a generic pattern for formulating a sub-goal for each quality property as follows:

(G5) The <system> is acceptably <quality property adjective>.

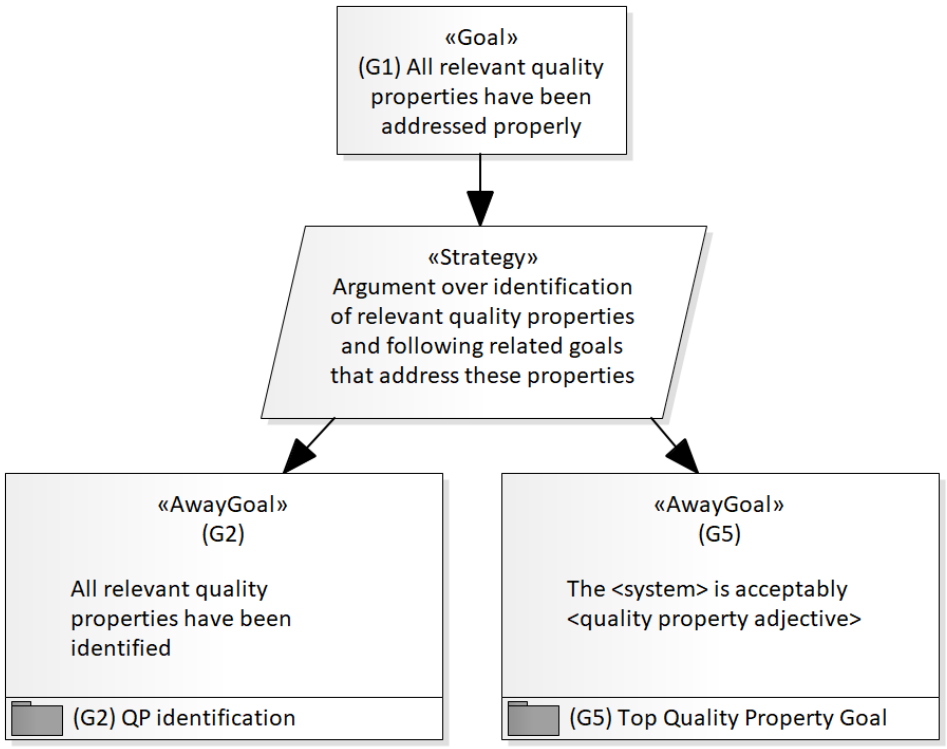


Figure 20 – Generic template for breaking down the top goal into sub-goals

To eliminate ambiguity – especially in complex system-of-systems – we recommend mentioning the system name already in the goal. The quality property adjective is an adjective related to the quality property to be addressed (e.g., secure, available, reliable). In Section 4.3.3, we will present a generic pattern for further refinement of quality goals.

4.3.2 Quality Property Identification

At the beginning of the use case elicitation, the quality properties that the system should satisfy must be defined. The stakeholders can name the quality properties that are relevant from their point of view. In order to facilitate the selection, a list of generally relevant quality properties in the form of a checklist is useful. In our case, we use the quality properties of ISO/IEC 25010, which have been defined for systems and software engineering. However, depending on the system to be developed, other quality properties may also be suitable. If a quality property is classified as relevant, this should be documented in a list of relevant quality properties.

If several stakeholders participate in a use case, all stakeholders should be asked about the relevant quality properties at the beginning of the analysis. In a subsequent step, use case elicitation can be continued. This ensures that the views of the other stakeholders are also considered in the subsequent process.

The procedure described here is illustrated in an assurance case as shown in Figure 21.

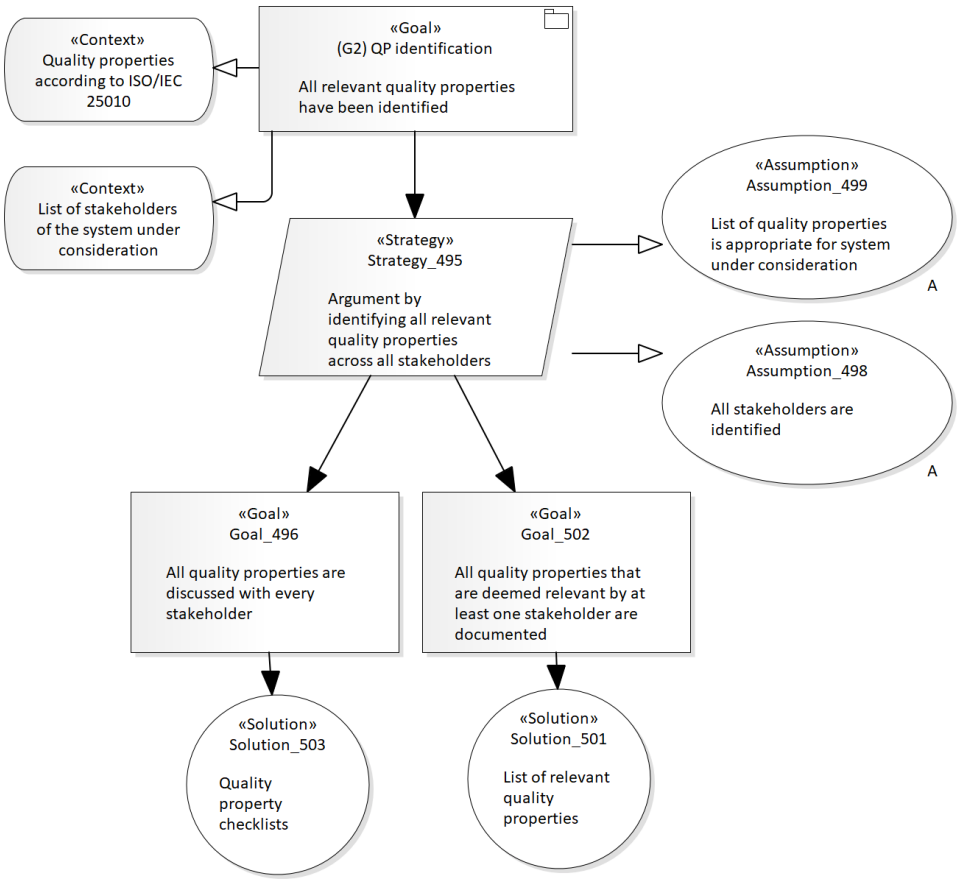


Figure 21 – Quality property identification argument

We introduced Goal_496 to ensure that all stakeholders are consulted. The completed quality property checklists serve as evidence. Goal_502 aims at documenting the relevant quality properties that appear important to at least one stakeholder. It is fulfilled by a list of relevant quality properties in Solution_501.

We assume here that the outgoing list of quality properties is appropriate for the system under consideration and that all stakeholders have been identified.

4.3.3 Generic Argumentation Pattern

For each quality property identified as relevant, a corresponding quality goal is now formulated that requires the fulfillment of the quality property. The generic pattern of Goal G5 shown in Figure 22 is suitable for this purpose. It will serve as the topmost quality property goal (e.g., “The system is acceptably safe”, “The system is acceptably secure”).

Since this formulation is still too abstract to implement or test the goal, it must be further decomposed into more specific goals. Our strategy is to derive concrete quality goals that are implementable and testable. We assume that the sum of the concrete quality goals completely fulfills the topmost quality property goal. For example, “The system is acceptably secure” is fulfilled when all security sub-goals are met.

The difficulty, however, lies in fully identifying a quality property’s sub-goals (Goal_444). Only then can the objectives be implemented (Goal_445) and finally verified (Goal_448).

That being said, the next issue to resolve is how to identify and formulate: quality goals. Figure 23 shows a proposal in which the *problems* that may occur in the context of the quality properties are identified first. Since not all problems have the same criticality, we focus the engineering effort on the more critical problems. Here, a risk-based approach has proven to be effective. In a risk assessment, the risks associated with the problems are evaluated. After the problems and their criticality have been determined, then, appropriate mitigating quality goals are formulated for each critical problem that prevent the occurrence of the problem or at least mitigate its consequences.

There are several ways to identify problems with regard to quality properties. In Section 4.4, we present an approach for systematically identifying problems using guidewords.

The fulfillment of Goal_458 can be documented in a risk assessment report. What the underlying assessment process and the resulting report looks like may vary with the quality property to be assessed. For the “safety” property, a Hazard and Risk Analysis Report, which will be introduced in Section 4.4, is most common.

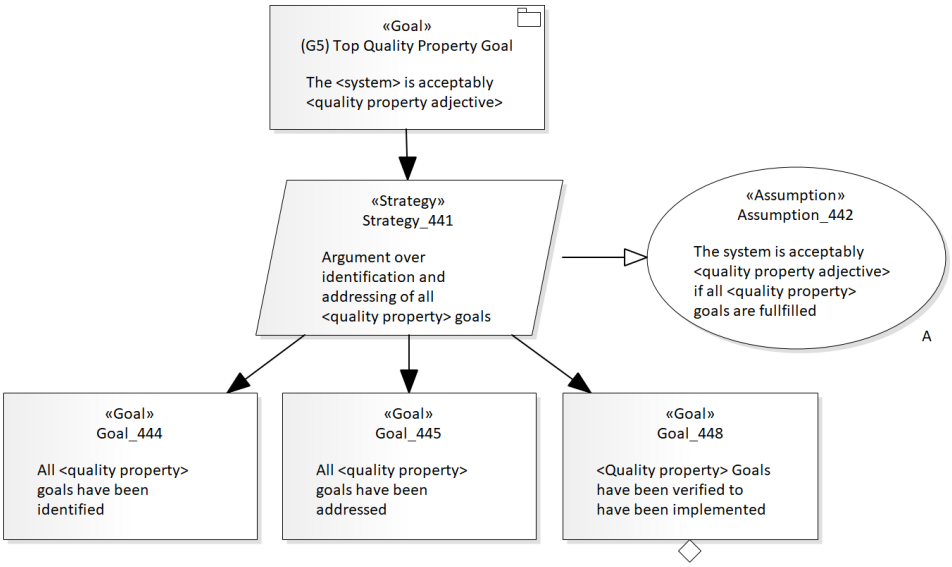


Figure 22 – First level of subdivision of a quality property goal

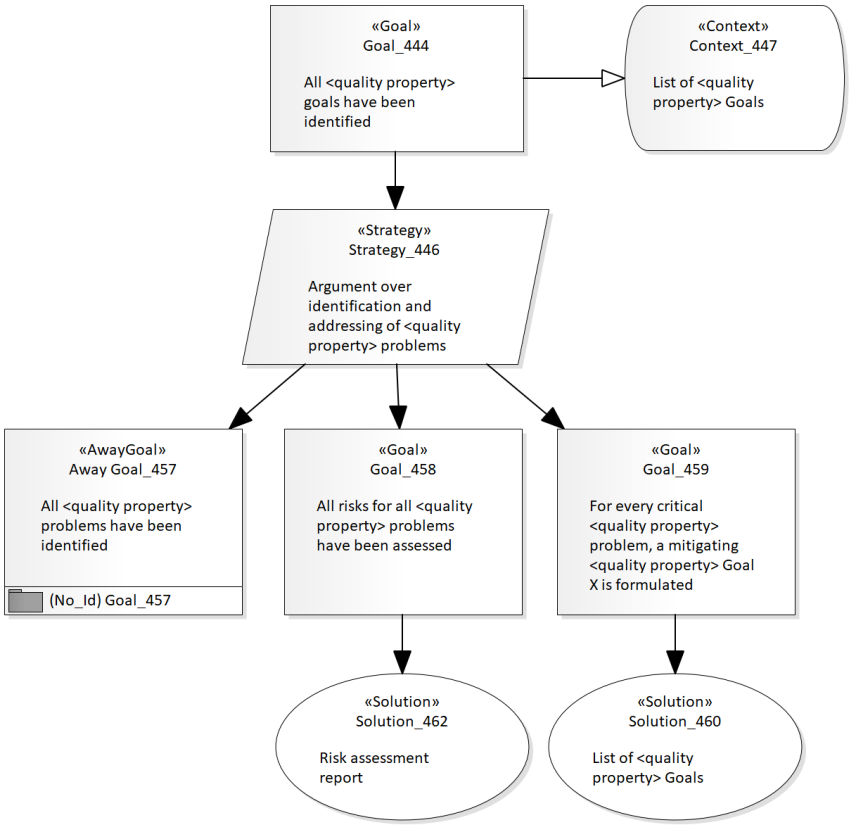


Figure 23 – Goal identification activity

Goal_459 addresses the formulation of mitigating goals for critical problems. The outcome of the goal identification activity is a list of quality goals that can be used to drive the development of a quality property. This is targeted by Goal_445. Figure 24 shows the refinement of this goal. As shown there, we assume that the quality goals are independent of each other in the sense that every single quality goal of the list could be considered separately.

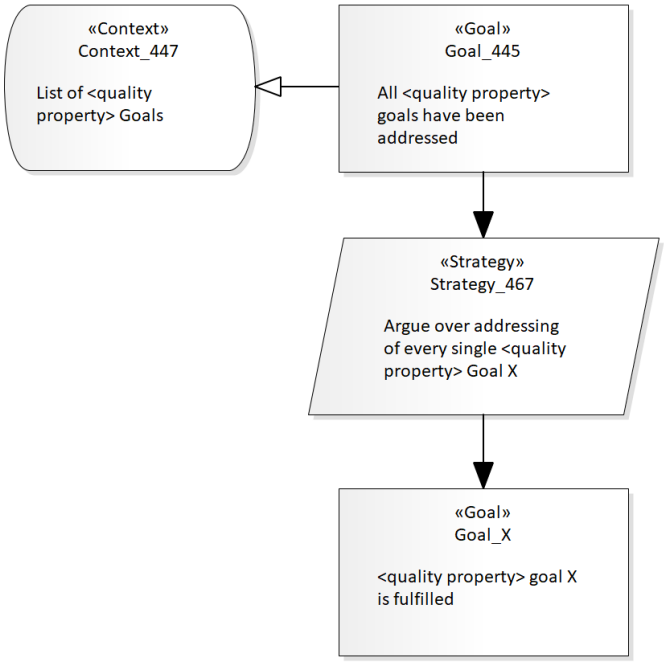


Figure 24 – Goal-addressing activity

When instantiating this generic pattern, all quality goals appear side by side at the position of Goal_X. Each goal is then elaborated individually according to the process shown in Figure 25. There, we assign one or more use case scenarios to each quality goal, each describing a system interaction where the quality is relevant.

The use case template (see Appendix A, UC Template Subsection 4.2) aims to detail scenarios as a stepwise description of the interaction between user and system (or peer system and system, respectively). Usually, the desired system behavior is described first, step by step, following the pattern sequence “user interacts with system”, “system responds to user”, and so on.

For each step in a scenario related to a quality goal, we can assess whether it contributes to the fulfillment of the quality goal or has the potential to violate the quality goal. In this sense, the information that is exchanged in a step (i.e., the output of the step) is of interest.

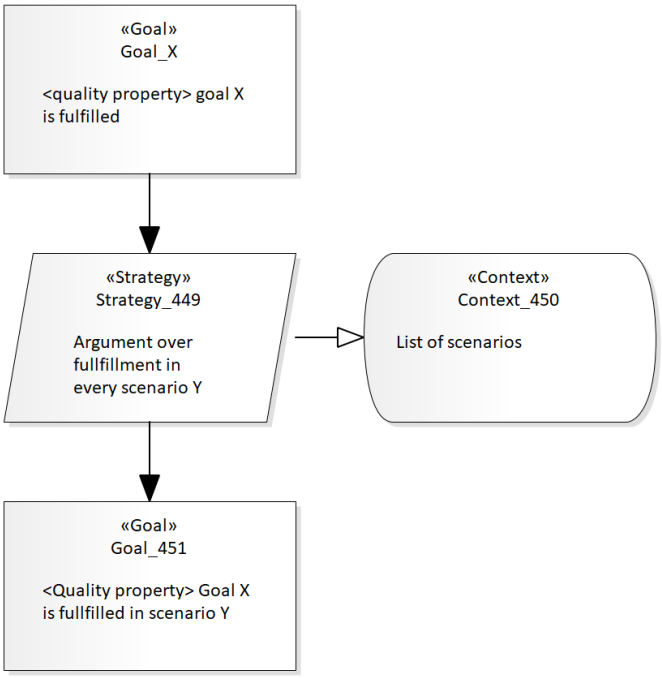


Figure 25 – Assignment of quality goals to use case scenarios

If a step potentially violates the quality goal, an alternative scenario is specified to describe the deviation of the desired behavior. Figure 26 shows the entire assessment process in the assurance case. The process ends with the reference to the alternative scenario.

Applying this generic argumentation pattern leads to a set of preliminary quality goals related to a quality property (e.g., security, safety, or availability). The use case template then contains scenario descriptions at the level of detail that is known at this stage of the systems engineering process. This information is passed on to subsequent engineering activities for further refinement.

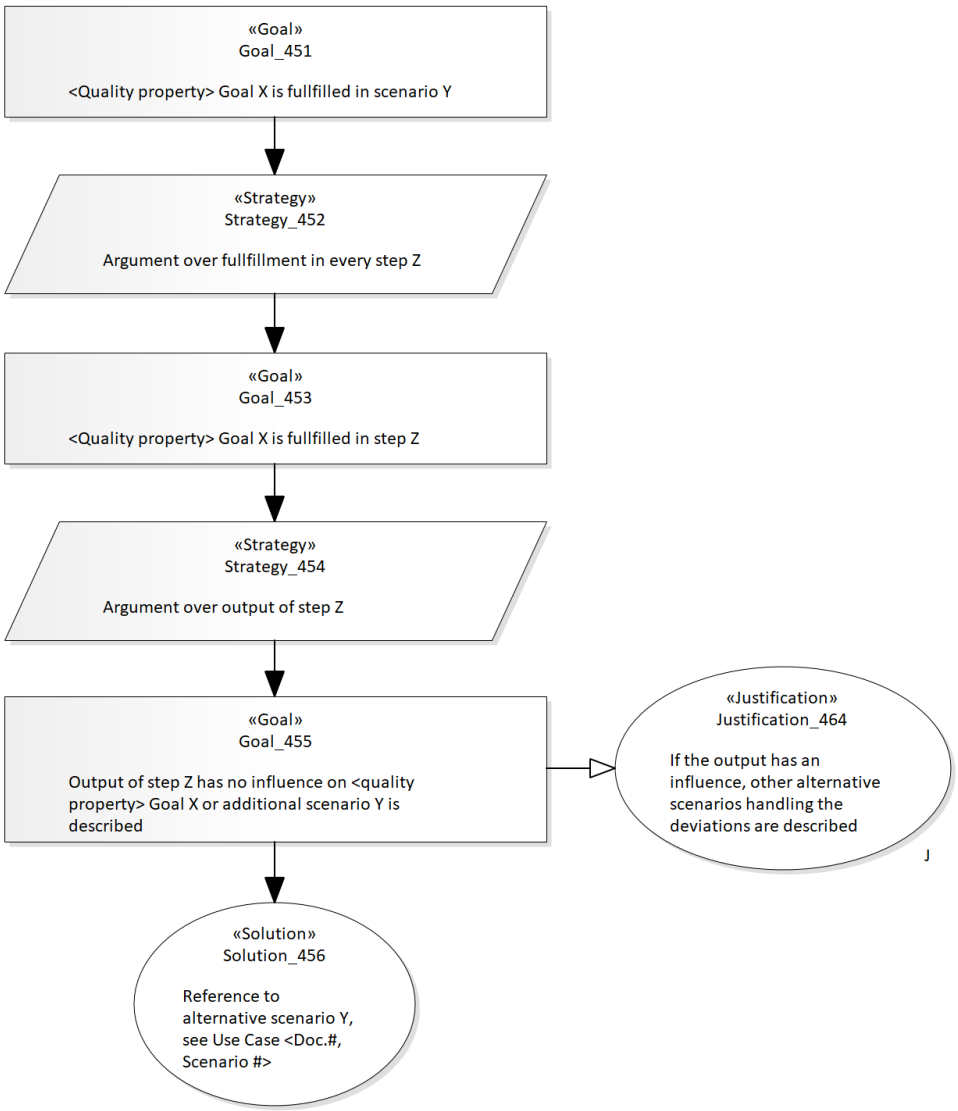


Figure 26 – Assessment of quality goal in scenario steps

4.4 Introduction to Problem Identification and Risk Assessment

To adequately address risks in terms of product quality, the conditions, causes, and specific effects of these risks must be determined first. Hazard Analysis and Risk Assessment (HARA) is a method for identifying such risks. Appropriate means for addressing them can then be specified accordingly [73, pp. 7-3]. The aim of HARA is to reduce the overall risk below a threshold at which the residual overall risk is considered acceptable or justifiable.

Across numerous application domains, HARA (or some equivalent risk analysis method) is highly recommended as an essential component for quality assurance. Examples include IEC 61508 [74], ISO 26262 [75], ARP 4754-A [76], or EN 50128 [77]. Hazard and Operability Studies (HAZOP), according to IEC 61882 [78] is a similar risk analysis method widely applied in the process industry.

This section provides general guidance on key elements of HARA application, as well as on non-safety quality risks. We will loosely adhere to the terminology established in ISO 26262 [75] for the automotive industry, but the concepts apply to related standards and other domains as well.

Central to HARA is the notion of a *Hazard*, an event that directly results in personal injury, property damage, or (outside the scope of ISO 26262 [75]) environmental damage. As an example of a hazard, consider a "vehicle breakdown". Transferred to other quality properties, we can speak here in generalized terms of a *Problem* (see Section 4.3.3 and Figure 23). In other words, we use the term *Problem* in the generic argumentation pattern, whereas a *Hazard* is the safety-specific instance of a problem.

Hazards can occur in a variety of scenarios, with each scenario being referred to as a Hazardous Event. Hazards are analyzed in different situations, since not every hazard posed by the system is equally critical in every situation. For instance, the hazardous event "vehicle breakdown while driving" is more critical than "vehicle breakdown while parking".

Causes of hazards can be internal or external to the system under development. Many safety standards restrict their analysis to internal causes; specifically, ISO 26262 focuses the analysis on the identification of deviations of key system elements from their intended behavior. These functional deviations are referred to as *Failure Modes*, and they specify how key system elements can fail. Failure modes are specified by HARA analysts, who typically do so on the basis of established *Guidewords* or application-specific judgment. For instance, a typical guideword is "No/Not", indicating the omission of a functionality of a system's element. Other typical guidewords are "Unintended", indicating a commission of functionality where it is not expected, or "More/Higher", "Less/Lower", "Early", and "Late" for value or timing problems. Depending on the type of system functionality and quality property under assessment, other guidewords could be appropriate, for instance "wrong sequence", "intermittent", or "reverse". In general, three flow types can be distinguished: "information flow", "material flow", and "energy flow". The analyst can combine the guideword with the functionality to formulate a failure mode and then use it for further analysis.

Once hazards have been identified, their severity, that is, the magnitude of the impact of their occurrence, and their likelihood of occurrence need to be evaluated. In principle, the combination of a hazard's severity and likelihood yields the hazard's risk.

In cases where these parameters are difficult to quantify accurately, qualitatively estimating the risk based on categorical risk criteria is typically preferred instead. In the case of ISO 26262 [75], risk parameters include "Exposure", "Severity", and "Controllability". *Exposure* reflects the likelihood of being an operating condition that may be hazardous if a particular failure mode occurs simultaneously. *Severity* reflects the impact of the hazard on people involved (which may include bystanders who are not directly involved in the operation of the system; for example pedestrians in the case of ISO 26262). *Controllability* is specific to ISO 26262 and reflects the ability of the driver or other people to avert the hazard during operation.

We propose using the HARA approach for each use case already in the early use case elicitation phase in order to identify related problems, to prioritize them, and formulate corresponding goals that address the problems. In the following, we will use an electric vehicle as our system under consideration to illustrate the approach.

We will take one of the use cases shown in Figure 27 to identify the various possible deviations of the use case. Beginning with "(UC1) Accelerate Vehicle", a potential failure mode is "No", resulting in loss of acceleration of the vehicle. The corresponding hazard (as an instance of a problem) could be classified as "vehicle breakdown". Table 14 shows a non-exhaustive example application of different failure modes to the use case.

In the next step, the identified problems are combined with relevant operational situations into which we refer to as problematic events. For instance, the hazard "vehicle breakdown" could be critical while driving through a tunnel. If a hazard is critical in a particular situation, the probability of the occurrence of that situation is important.

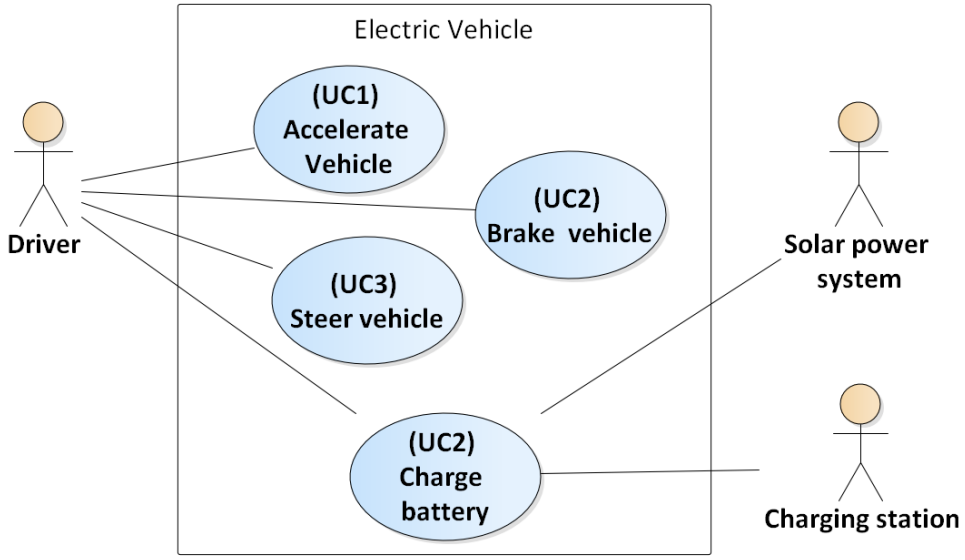


Figure 27 – Example use cases of an electric vehicle

In practice, it is often difficult to determine reasonably accurate quantitative values for the probability. Therefore, a qualitative approach with different probability classes is a pragmatic and lightweight solution.

Table 14 – Example Functional Problem Analysis (FPA)

FPA ID	UC ID	Use Case	Failure Mode	Malfunction / Deviation	Problem
FPA_1	UC1	Accelerate Vehicle	No/Not	Vehicle does not accelerate	H_1: Vehicle breakdown
FPA_2	UC1	Accelerate Vehicle	Unintended	Vehicle accelerates unintentionally	H_2: Unintended vehicle acceleration

The occurrence probability should be classified in accordance with the domain-specific definitions of the corresponding standards, where available. In Table 15, different situations are assigned to our exemplary hazards, and the exposure classes of ISO 26262-3 [5] are used (see Table 22 in Appendix B) to characterize their likelihood. If no classes are defined in a standard, Table 16 may be used instead. We extended the classification to a simple semi-quantitative approach to allow mathematical operations on the classes when combining them with other metrics. The four classes “Highly unlikely”, “Low probability”, “Medium probability”, and “High probability” enable a coarse-grained classification of the duration or frequency of exposure.

Table 15 – Example analysis of hazards in specific situations

Problematic Event ID	Problem ID	Problem	Situation Description	Exposure	Argument
HE_1	H_1	Vehicle breakdown	Vehicle driving through tunnel	E2 – low probability	Situation occurs statistically less than 1% of the operating time (0.4 h/a … 4 h/a)
HE_2	H_2	Unintended vehicle acceleration	Vehicle driving on highway	E4 – high probability	Determined by expert assessment

The proposed procedure demonstrated in Table 15 was intended to be lightweight in order to enable quick and easy assessment exposure. Where more information about probability values or exposure times is available, this should already be documented in the column “Argument” and passed on to subsequent engineering activities.

Table 16 – Semi-quantitative exposure classification

Class	E0 = 0	E1 = 1	E2 = 2	E3 = 3
Description	Highly unlikely	Low probability	Medium probability	High probability

In Table 20, all risk factors related to the problematic event – i.e., situation exposure, severity of consequence, and controllability – have been assigned. In this example, the metrics of ISO 26262-3 [5] (see Table 22, Table 23, and Table 24 in Appendix B) are used. If no other metric is available, the general-purpose classifications introduced in Table 17 and Table 18 may be used instead.

Table 17 – Semi-quantitative severity classification

Class	S0 = 0	S1 = 1	S2 = 2	S3 = 3
Description	No loss	Low severity	Medium severity	High severity

Table 18 – Semi-quantitative controllability classification

Class	**C0 = 0**	**C1 = 1**	**C2 = 2**	**C3 = 3**
Description	Controllable in general	High controllability	Normal controllability	Low controllability

To express the overall risk of a problematic event, we propose combining all factors into one central metric by multiplying them. We further refer to this as criticality metric (CM):

$$CM = E * S * C$$

CM can be used to prioritize the use cases and quality goals. Another important aspect of CM is the associated thoroughness that is required in subsequent analysis. A complete overview of resulting values is shown in Table 19.

It should be noted that this is just a simple utility to allow the requirements engineer to make a quick and rough assessment of the criticality. Therefore, not too much importance should be attached to the individual values. Equal values correspond to similar criticality, even tendencies between two values (larger or smaller) are possible, but further mathematical operations have no semantics. For example, twice the CM does not necessarily mean twice the criticality. The same limitations exist here as with the Risk Priority Number (RPN) of the Failure Mode and Effects Analysis (FMEA) [79] and other similarly designed metrics.

The multiplications by zero (E0, S0, or C0) are intentional and lead directly to a CM of zero, regardless of the values of the other parameters. These problems or derived quality goals can probably be directly neglected in most cases, either because they do not have a large impact by themselves, or they are usually controllable by the user, or they are only relevant in situations, which are very unlikely to occur.

Table 19 – Criticality Metric Matrix

Sever-ity class	Expo-sure class	Controllability class			
		0	**1**	**2**	**3**
0	**0**	0	0	0	0
0	**1**	0	0	0	0
0	**2**	0	0	0	0
0	**3**	0	0	0	0
1	**0**	0	0	0	0
1	**1**	0	1	2	3
1	**2**	0	2	4	6
1	**3**	0	3	6	9
2	**0**	0	0	0	0
2	**1**	0	2	4	6
2	**2**	0	4	8	12
2	**3**	0	6	12	18
3	**0**	0	0	0	0
3	**1**	0	3	6	9
3	**2**	0	6	12	18
3	**3**	0	9	18	27

Instead of CM, ISO 26262-3 [5] proposes a conversion table that relates the factors to the automotive-specific metric of Automotive Safety Integrity Levels (ASIL). For instance, the hazardous event “Vehicle breakdown while driving through tunnel” (HE_1), which is rated E2, S3, and C2, results in ASIL A according to Table 25 in Appendix B.

Table 20 – Example assessment of problematic events

Problematic Event ID	Problematic Event Description	Consequence	Exposure	Severity	Controllability
HE_1	Vehicle breakdown while driving through tunnel	Rear-end collision caused by following vehicle. Tunnel road blocked.	E2 – low probability	S3 – fatal	C2 – normal
HE_2	Unintended vehicle acceleration on highway	Frontal collision with another vehicle	E4 – high probability	S2 – severe	C2 – normal

The loop to the assurance case is closed by deriving quality goals as a system objective to respond to a problematic event. The system could respond either by reducing the severity or the probability of the problematic event, or by improving its controllability. In our case, the safety goal "Vehicle shall not break down without warning the driver within 500 msec. Safe state: Warning issued" addresses HE_1 by reducing the probability of an actual accident through higher controllability by the driver, who can decide on further actions (e.g., maneuver, hazard lights on, etc.). Table 21 contains the full description of our example hazardous events.

Table 21 – Quality goal assignment

Problematic Event ID	Problematic Event Description	Criticality Level	Quality Goal ID	Quality Goal
HE_1	Vehicle breakdown while driving through tunnel	ASIL A	SG_1	Vehicle shall not break down without warning the driver within 500 msec. Safe state: Warning issued
HE_2	Unintended vehicle acceleration on highway	ASIL B	SG_2	Vehicle shall not accelerate unintentionally for longer than 200 msec. Safe state: Propulsion off; warning issued

The generic assurance case from Section 4.3 targets Goal_457, that is, the complete identification of all problems. Figure 28 illustrates the further breakdown to achieve this goal by following a systematic process.

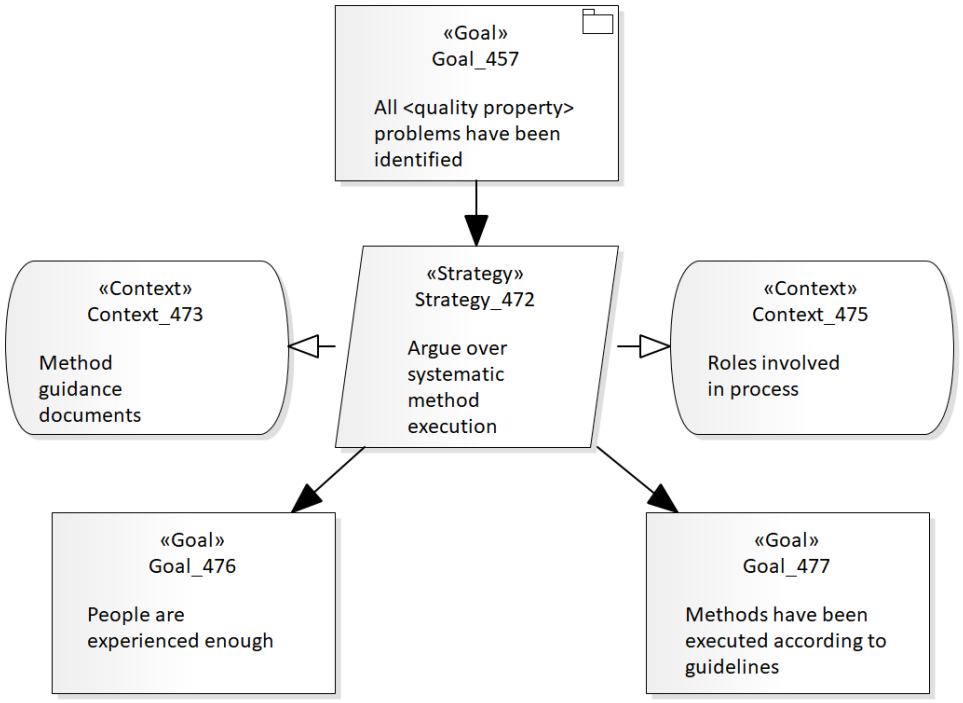


Figure 28 – Problem identification argument

The systematic process is based on step-by-step guidelines, as described here with the HARA approach. Even though we tried to provide a proper description of the process steps, the presence of people with experience in the HARA method as well as stakeholders who are familiar with the problems that might occur in the context of a system is essential. Therefore, the process shown in Figure 29 requires at least reflection about the people involved in the analysis and, if possible, evidence for their proficiency should be provided, e.g., training certificates or a convincing curriculum vitae.

Apart from experienced people, the underlying method must be mature and needs to be followed according to the guidelines required in Goal_477 (see Figure 28 and Figure 30). This could be proven by a (short) method analysis with respect to the appropriateness of the guidewords used and the systematic application of the guidewords to each use case. Reviews and a proper versioning approach promote proper execution of the method.

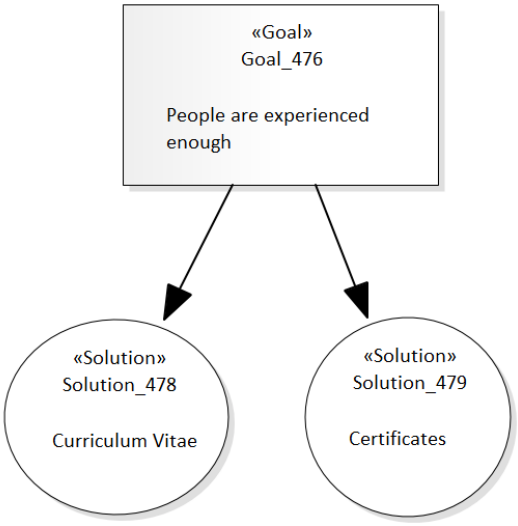


Figure 29 – Evidence of experience

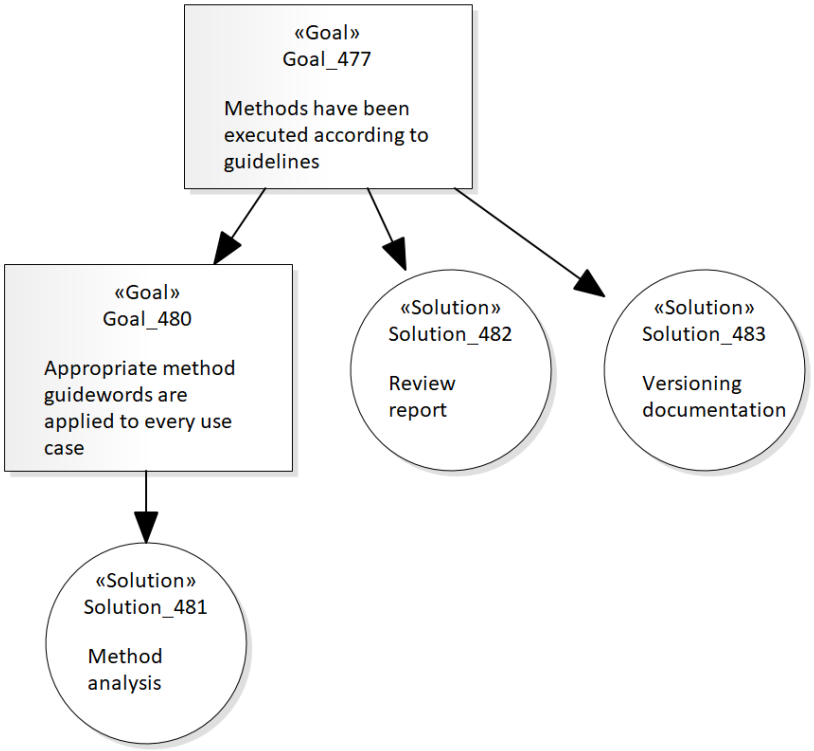


Figure 30 – Evidence of proper problem identification process

5 Combining Use Cases and Assurance Cases

Tool support ensures consistency of the information in the use case description and in the assurance case. It maintains the linkage between the different parts of the use case description and the respective elements in the assurance case model.

5.1 Linkage Between Objectives

In Figure 31, we show the linkage between the use case objective mentioned in the use case template and the corresponding objective (referred to as Goal) in the assurance case. Additional rationale for the goal can be provided in the assurance case as well. In the complete model (see Section 4.3.3), these objectives are systematically addressed using a top-down approach.

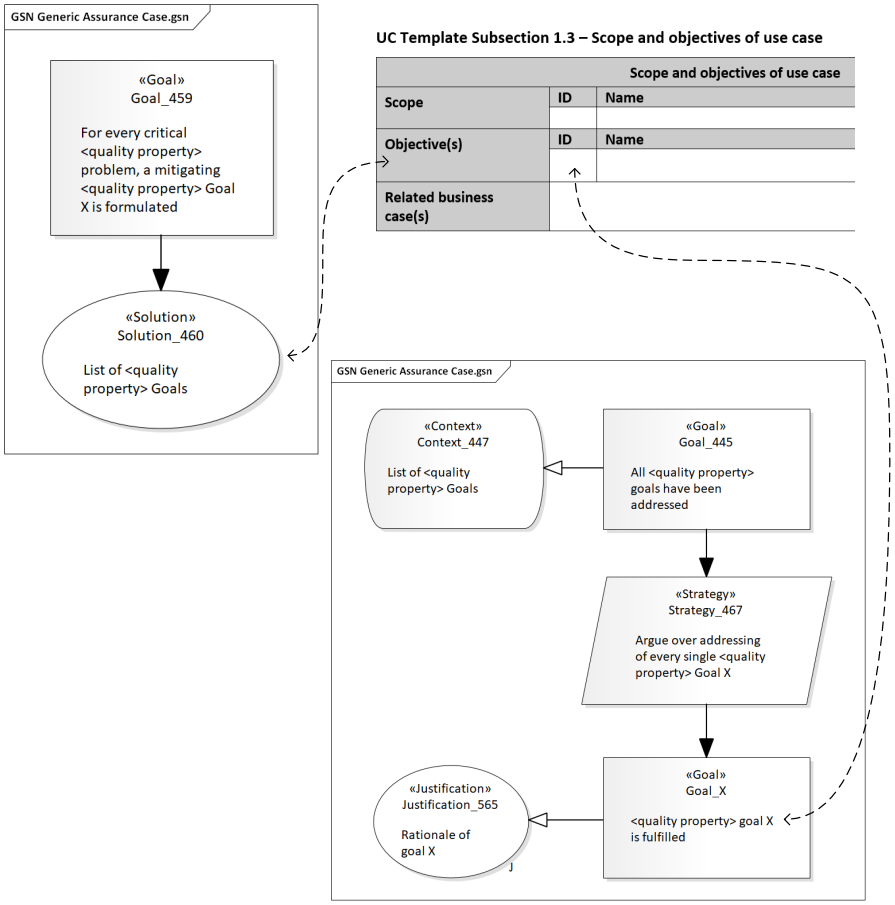


Figure 31 – Objectives

5.2 Linkage Between Scenarios

For each objective, we maintain a link between the use case scenarios, mentioned in the use case template and the corresponding scenarios in the assurance case, as illustrated in Figure 32.

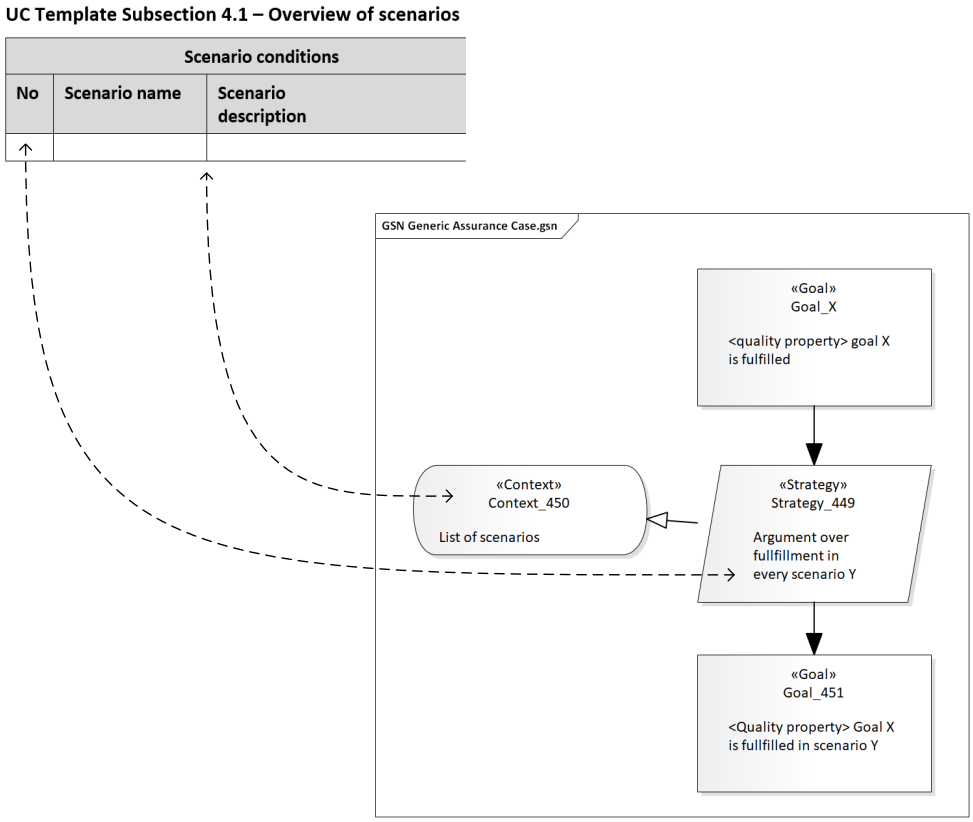


Figure 32 – Scenarios

In the completed model (see Section 4.3.3, Figure 26), every problem defined in these scenarios is systematically mitigated.

5.3 Linkage Between Steps

In Figure 33, we show the linkage between the steps of every scenario mentioned in the use case template and the corresponding steps in the assurance case.

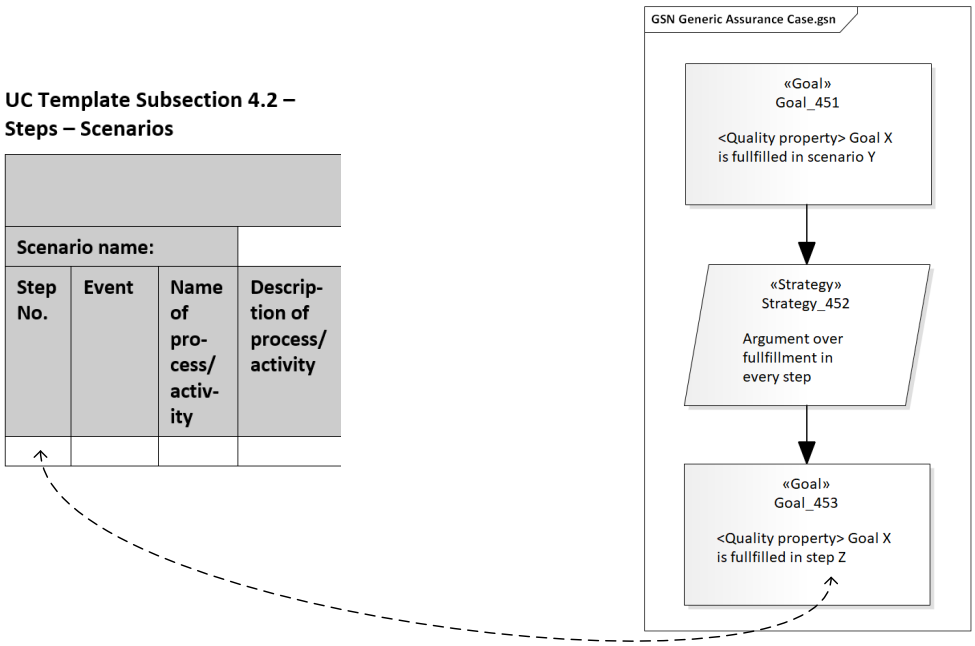
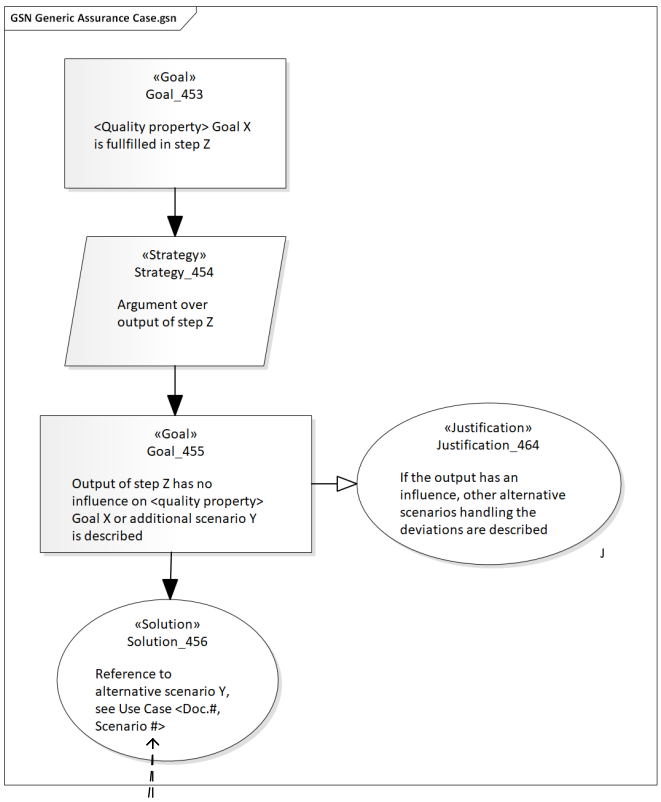


Figure 33 – Steps

In the complete model (see Section 4.3.3, Figure 26), every step is validated against the underlying goal.

5.4 Linkage Between Alternative Scenarios



UC Template Subsection 4.1 – Overview of scenarios

Scenario conditions		
No	**Scenario name**	**Scenario description**

UC Template Subsection 4.2 – Steps – Scenarios

Scenario										
Scenario name:										
Step No.	**Event**	**Name of process/ activity**	**Description of process/ activity**	**Service**	**Information producer (actor)**	**Information receiver (actor)**	**Information exchanged (IDs)**	**Requirements R-ID**	**Assumption ID**	**Alternative scenario ID**

Figure 34 – Alternative scenario reference

In Figure 34, we illustrate the relation between different use case scenarios mentioned in the use case template and the corresponding relation in the assurance case. In the complete model (see Section 4.3.3, Figure 26), if a specific step of a scenario affects the underlying goal, then another scenario (of the same use case) is referenced describing the possible deviations.

6 Summary

In the SINTEG project enera, challenges and appropriate solutions were researched that will accompany us on the way to a Smart Grid. Smart Grids clearly represent a socio-technical system of systems due to their structure of multiple integrated complex systems with pluralistic goals and strong interactions with user behavior.

In this work, we summarize the special characteristics and challenges of complex systems and systems of systems. Based on these characteristics, we present a possible Systems-of-Systems Engineering (SoSE) process that has to fulfill eight essential requirements:

- Requirement 1 (Establish shared understanding),
- Requirement 2 (Enable collaboration),
- Requirement 3 (Ensure interoperability),
- Requirement 4 (Ensure confidentiality of information),
- Requirement 5 (Visualize all relevant information simultaneously),
- Requirement 6 (Enable continuous monitoring),
- Requirement 7 (Enable integration into existing models),
- Requirement 8 (Use purposeful tools and methodologies).

The SoSE process outlined in this book consists of the creation of four submodels:

- the Artifact Model, consisting of the artifacts “Shared Understanding”, “Body of Knowledge”, “Ubiquitous Language”, and “Standards and Standardization Gaps”;

- the Activity Model, consisting of the activities “Establish a shared understanding”, “Identify standards”, and “Updating the documentation”;

- the Role Model, consisting of the roles “Requirements Owner”, “System Designer”, “System Support”, and “Coordinator”, and finally

- the Sequence Model, which describes the chronological order of the activities and defines milestones after every activity.

Furthermore, we present some tools and methodologies that can be used in the process to address the process requirements and elaborate elements of the submodels, such as Lego Serious Play, Reference Architecture Models (RAM), and use cases.

Reference Architecture Models (RAM) are used for the visual classification of *logical system elements*. They provide different views of a system at different levels of abstraction.

To determine the *logical system elements*, we propose use cases. As use cases are used especially in early systems engineering phases, knowledge about the system under consideration is limited during use case specification, especially with regard to system architecture and system assets (information assets, process assets, and physical assets). To bridge this temporary gap, use cases can describe the system behavior from the user's point of view even with limited knowledge about the system architecture and its elements.

During requirements elicitation with use cases, stakeholders primarily describe functional properties of the system to be developed. In order to bring non-functional requirements into the focus of the stakeholders, we propose explicitly directing their attention towards non-functional system qualities and working through them systematically. To support this, we have developed a guideline that helps to formulate and track the desired quality goals. As documentation of the guideline, we introduce assurance cases with the Goal Structuring Notation (GSN).

By reducing the problem of quality assurance to proper coverage of isolated quality properties, general quality goals could be traced down to individual quality requirements. If already present in the early system development phase, the trace also reaches to solution elements that implement the requirements and therefore fulfill the goal.

The top goal of the presented assurance case is to properly address all quality properties that are deemed relevant for the stakeholders. In order to identify the *relevant* quality properties from the list of potential system quality properties, we presented a corresponding process for this purpose, which assures that all stakeholders have been thought of for all quality properties and that corresponding quality goals are formulated.

We also provide a process for addressing the identified quality goals. They cover the assessment of each quality goal in the use case scenarios as well as their step-by-step description. Whenever a step has the potential to violate the currently considered quality goal, the proposed process demands an additional scenario describing the system's reaction to possible deviation(s) from the desired outcome.

For systematic risk analysis, we provide a problem identification method based on guidewords applied to use case descriptors. Furthermore, a semi-quantitative risk assessment is proposed, which considers problems and, where appropriate, problematic operational situations. We propose using this approach already in the early

use case elicitation phase for each use case to identify related problems, prioritize them, and formulate corresponding goals that address the problems.

Finally, we propose utilizing the IEC 62559 use case template and instantiating the presented generic argumentation pattern in parallel during use case elicitation with the stakeholders. Linkage between use case template elements and assurance case elements should be established in both directions. The assurance case can provide additional information about strategy, context, or assumptions that is not explicitly mentioned in the use case template. Moreover, the assurance case enables complete traceability across all related elements. Regarding the entire SoSE process, it looks promising to visualize as many elements of the engineering process as possible and to manage them in a tool-supported model in order to switch between different views.

7 References

[1] Bundesministerium für Wirtschaft und Energie, "enera – Digitalisation of energy supply," 2020. [Online]. Available: https://www.sinteg.de/en/showcases/enera/. [Accessed 25 August 2020].

[2] IEC, "IEC 62559-2 - Use case methodology - Part 2: Definition of the templates for use cases, actor list and requirements list," 2015.

[3] International Electrotechnical Commission, "IEC SRD 62913-1:2019 Generic smart grid requirements – Part 1: Specific application of the Use Case methodology for defining generic smart grid requirements according to the IEC systems approach," 2019.

[4] International Organization for Standardization/International Electrotechnical Commission, "ISO/IEC 25010:2011 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models," 2011.

[5] International Organization for Standardization, "ISO 26262-3:2018 - Road vehicles — Functional safety — Part 3: Concept phase," 2018.

[6] Bibliographisches Institut GmbH, "Duden - System," [Online]. Available: https://www.duden.de/rechtschreibung/System. [Accessed 31 August 2020].

[7] L. von Bertalanffy, General system theory: foundations, development, applications, G. Braziller, 1968.

[8] J. Boardman and B. Sauser, Systems Thinking: Coping with 21st Century Problems, vol. Systems Innovation Book Series, CRC Press, 2008.

[9] H. Sillitto, R. M. Griego, S. Jackson, D. Krob, P. Godfrey, E. Arnold, J. Martin and D. McKinney, “Defining “System”: a Comprehensive Approach,” in *INCOSE International Symposium*, Adelaide (Australia), 2017.

[10] H. Sillitto, J. Martin, D. Mckinney, R. Griego, D. Dori, D. Krob, P. Godfrey, E. Arnold and S. Jackson, "Systems Engineering and System Definitions," 2019.

[11] SEBoK Authors, "Guide to the Systems Engineering Body of Knowledge (SEBoK)," 15 Mai 2020. [Online]. Available: https://www.sebokwiki.org/wiki/Guide_to_the_Systems_Engineering_Body_of_Knowledge_(SEBoK). [Accessed 23 10 2019].

[12] International Organization for Standardization/International Electrotechnical Commission/Institute of Electrical and Electronics Engineers, "ISO/IEC/IEEE 21839:2019 - Systems and software engineering — System of systems (SoS) considerations in life cycle stages of a system," 2019.

[13] M. Maier, "Architecting principles for systems-of-systems," *Systems Engineering,* vol. 1, no. 4, pp. 267-284, 1998.

[14] B. Sauser and J. Boardman, "Taking hold of system of systems management," *EMJ - Engineering Management Journal,* vol. 20, no. 4, 2008.

[15] J. Boardman and B. Sauser, "System of Systems - The meaning of of," in *Proceedings 2006 IEEE/SMC International Conference on System of Systems Engineering*, Los Angeles, CA, USA, 2006.

[16] W. C. Baldwin and B. Sauser, "Modeling the characteristics of system of systems," in *2009 IEEE International Conference on System of Systems Engineering (SoSE)*, Albuquerque, NM, 2009.

[17] International Organization for Standardization/International Electrotechnical Commission/Institute of Electrical and Electronics Engineers, "ISO/IEC/IEEE 21841:2019 - Systems and software engineering — Taxonomy of systems of systems," 2019.

[18] J. S. Dahmann and K. J. Baldwin, "Understanding the current state of US defense systems of systems and the implications for systems engineering," in *2008 IEEE International Systems Conference Proceedings, SysCon 2008*, Montreal, Canada, 2008.

[19] International Organization for Standardization, *ISO/IEC/IEEE 21841:2019 - Systems and software engineering — Taxonomy of systems of systems,* Geneva, Switzerland, 2019.

[20] NASA, NASA System Engineering Handbook Revision 2, 2016.

[21] C. Ncube and S. L. Lim, "On systems of systems engineering: A requirements engineering perspective and research agenda," in *2018 IEEE 26th International Requirements Engineering Conference*, Banff, Alberta, Canada.

[22] Office of the Deputy Under Secretary of Defense for Acquisition and Technology, Systems and Software Engineering. Systems Engineering Guide for Systems of Systems, 1 ed., Washington, DC: ODUSD(A&T)SSE, 2008.

[23] R. L. Ackoff, "Towards a System of Systems Concepts," vol. 17, no. 11, 1971.

[24] H. A. Simon, The sciences of the artificial, MIT Press, 1996.

[25] N. B. Shah, D. H. Rhodes and D. E. Hastings, "Systems of Systems and Emergent System Context," in *Proceedings of the 5th Annual Conference on Systems Engineering Research*, 2007.

[26] D. Hess, *Heterogeneous and homogeneous groups in the innovation process,* 2007 ed., P. D. T. Blanke, P. D. M. Heidenreich and P. D. H. Trautwein, Eds., 2007.

[27] P. R. Carlile, "Transferring, translating, and transforming: An integrative framework for managing knowledge across boundaries," *Organization Science,* vol. 15, no. 5, 2004.

[28] C. E. Shannon and W. Weaver, A Mathematical Theory of Communication, Illinois, United States: University of Illinois Press, 1963.

[29] P. R. Carlile, "A pragmatic view of knowledge and boundaries: Boundary objects in new product development," *Organization Science,* vol. 13, no. 4, 2002.

[30] D. A. Broniatowski and C. L. Magee, "The Emergence and Collapse of Knowledge Boundaries," *IEEE Transactions on Engineering Management,* 2017.

[31] A. M. Madni and M. Sievers, "System of systems integration: Key considerations," *Systems Engineering,* vol. 17, no. 3, 2014.

[32] H. Van Der Veer and A. Wiles, *Achieving Technical Interoperability: the ETSI Approach,* European Telecommunications Standards Institute, 2008.

[33] J. Dahmann, "System of Systems Pain Points," *INCOSE International Symposium,* vol. 24, no. 1, pp. 108-121, 2014.

[34] D. Rost, M. Naab, C. Lima and C. Von Flach Garcia Chavez, "Software architecture documentation for developers: A survey," in *Software Architecture. ECSA 2013. Lecture Notes in Computer Science*, vol. 7957, Springer, Berlin, Heidelberg, 2013.

[35] T. C. Lethbridge, J. Singer and A. Forward, "How software engineers use documentation: the state of the practice," *IEEE Software,* vol. 20, no. 6, pp. 35-39, 2003.

[36] International Organization for Standardization/International Electrotechnical Commission/Institute of Electrical and Electronics Engineers, "ISO/IEC/IEEE 29148:2011 - Systems and software engineering — Life cycle processes — Requirements engineering," 2011.

[37] C. Keating, J. J. Padilla and K. Adams, "System of systems engineering requirements: Challenges and guidelines," *EMJ - Engineering Management Journal,* vol. 20, no. 4, pp. 24-31, 2008.

[38] H. W. J. Rittel and M. M. Webber, "Dilemmas in a general theory of planning," *Policy Sciences,* vol. 4, no. 2, pp. 155-169, 1973.

[39] M. Broy and M. Kuhrmann, Projektorganisation und Management im Software Engineering, Springer Vieweg, 2013.

[40] P. R. Smart, D. Mott, K. Sycara, D. Braines, M. Strub and N. R. Shadbolt, "Shared Understanding within Military Coalitions: A Definition and Review of Research Challenges," in *Knowledge Systems for Coalition Operations*, Southampton, United Kingdom, 2009.

[41] E. A. C. Bittner and J. M. Leimeister, "Why shared understanding matters - Engineering a collaboration process for shared understanding to improve collaboration effectiveness in heterogeneous teams," in *Proceedings of the Annual Hawaii International Conference on System Sciences*, Hawaii, 2013.

[42] J. Aranda, *A Theory of Shared Understanding for Software Organizations,* Toronto, Canada, 2010.

[43] E. Evans, Domain-driven design: tackling complexity in the heart of software, Addison-Wesley, 2004.

[44] H. Tenzer and M. Pudelko, "The impact of language barriers on shared mental models in multinational teams," in *Academy of Management 2012 Annual Meeting, AOM 2012*, Boston, MA, 2012.

[45] L. R. Hoffman and N. R. F. Maier, "Quality and acceptance of problem solutions by members of homogeneous and heterogeneous groups," *Journal of Abnormal and Social Psychology,* vol. 62, no. 2, p. 401–407, 1961.

[46] U. Hammerschall, *Flexible Methodenintegration in anpassbare Vorgehensmodelle,* 2008.

[47] S. A. Sheard, "Twelve System Engineering Roles," *INCOSE International Symposium,* vol. 6, no. 1, pp. 478-485, 1996.

[48] O. Linssen and M. Kuhrmann, "Vorgehensmodelle für das Projektmanagement," in *Basiswissen Projektmanagement - Grundlagen der Projektarbeit*, Symposion Publishing GmbH, 2013, pp. 153-188.

[49] E. Frick, S. Tardini and L. Cantoni, *White Paper on LEGO®SERIOUS PLAY® - its applications in Europe,* U. d. S. italiana, Ed., Lugano, Switzerland, 2013, p. 29.

[50] P. Kristiansen and R. Rasmussen, Building a Better Business Using the Lego Serious Play Method, Hoboken, New Jersey: John Wiley & Sons, Inc, 2014.

[51] S. Blair and M. Rillo, Serious Work: How to Facilitate Meetings & Workshops Using the Lego Serious Play method, ProMeet, 2016.

[52] M. Uslar and S. Hanna, "Model-driven Requirements Engineering Using RAMI 4.0 Based Visualizations," in *Modellierung 2018 - AQEMO: Adequacy of Modeling Methods*, Braunschweig, 2018.

[53] M. Gottschalk, M. Uslar and C. Delfs, The Use Case and Smart Grid Architecture Model Approach, vol. Springer Briefs in Energy, Springer International Publishing, 2017.

[54] M. Clausen, M. Gottschalk, S. Hanna, C. Kronberg, C. Rosinger, M. Rosinger, J. Schulte, J. Schütz and M. Uslar, "Smart Grid Security Method: Consolidating Requirements Using a Systematic Approach," in *CIRED WORKSHOP 2018 proceedings "Microgrids and local energy communities"*, Ljubljana, Slovenia, 2018.

[55] M. Uslar, S. Rohjans, C. Neureiter, F. Pröstl Andrén, J. Velasquez, C. Steinbrink, V. Efthymiou, G. Migliavacca, S. Horsmanheimo, H. Brunner and T. Strasser, "Applying the Smart Grid Architecture Model for Designing and Validating System-of-Systems in the Power and Energy Domain: A European Perspective," *Energies,* vol. 12, no. 2, p. 258, 2019.

[56] B. Weinert, M. Uslar and A. Hahn, "System-of-systems: How the maritime domain can leam from the Smart Grid," in *2017 International Symposium ELMAR*, Zadar, Croatia, 2017.

[57] R. Santodomingo, M. Uslar, M. Gottschlak, A. Goering, L. Nordström and G. Valdenmaiier, "The DISCERN tool support for knowledge sharing in large Smart Grid projects," in *CIRED Workshop 2016*, Helsinki, Finland , 2016.

[58] M. Uslar and D. Engel, "Towards Generic Domain Reference Designation: How to learn from Smart Grid Interoperability," in *D-A-Ch Energieinformatik 2015*, Karlsruhe, 2015.

[59] GridWise Architecture Council, "Smart Grid Interoperability Maturity Model Summary," [Online]. Available: https://www.gridwiseac.org/about/imm.aspx. [Accessed 16 08 2020].

[60] CEN/CENELEC/ETSI Smart Grid Working Group Reference Architecture, "Reference Architecture for the Smart Grid," 2012.

[61] International Organization for Standardization/International Electrotechnical Commission, "ISO/IEC 19505-2:2012 - Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 2: Superstructure," 2012.

[62] International Electrotechnical Commission, "IEC 62559-2:2015 - Use case methodology - Part 2: Definition of the templates for use cases, actor list and requirements list," 2015.

[63] T. Munzner, Visualization Analysis and Design, CRC Press, 2014.

[64] R. Koschke, "Software visualization in software maintenance, reverse engineering, and re-engineering: A research survey," *Journal of Software Maintenance and Evolution,* vol. 15, no. 2, 2003.

[65] T. Panas, T. pperly, D. Quinlan, A. Sæbjørnsen and R. Vuduc, "Communicating software architecture using a unified single-view visualization," in *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, ICECCS*, 2007.

[66] T. P. Kelly, "Arguing Safety – A Systematic Approach to Managing Safety Cases," University of York, York, UK, 1998.

[67] SCSC Assurance Case Working Group, "Goal Structuring Notation Community Standard Version 2," York, 2018.

[68] Adelard LLP, "CAE FRAMEWORK," Adelard LLP, 2020. [Online]. Available: https://claimsargumentsevidence.org/. [Accessed 18 August 2020].

[69] Object Management Group, "Structured Assurance Case Metamodel v2.1," OMG, 2020.

[70] International Organization for Standardization/International Electrotechnical Commission, "ISO/IEC 25012:2008 - Software engineering – Software Product Qualiy Requirements and Evaluation 8SQuaRE) – Data quality model," 2008.

[71] International Electrotechnical Commission, "IEC PAS 62559:2008 - IntelliGrid methodology for developing requirements for energy systems (Withdrawn)," 2008.

[72] enera Konsortium, "enera Projektkompendium," EWE Aktiengesellschaft, Oldenburg, Germany, 2021.

[73] U.S. Department of Transportation - Federal Aviation Administration, System Safety Handbook, Washington DC, USA, 2000.

[74] International Electrotechnical Commission, "IEC 61508:2010 - Functional safety of electrical/electronic/programmable electronic safety-related systems," 2010.

[75] International Organization for Standardization, "ISO 26262:2018 - Road vehicles — Functional safety," 2018.

[76] SAE International, "ARP4754A - Guidelines for Development of Civil Aircraft and Systems," SAE International, Warrendale, PA, United States, 2010.

[77] DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE, "DIN EN 50128; VDE 0831-128:2012-03 - Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems; German version EN 50128:2011," 2012.

[78] International Electrotechnical Commission, "IEC 61882:2016 - Hazard and operability studies (HAZOP studies) - Application guide," 2016.

[79] DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE, "DIN EN 60812 - Analysetechniken für die Funktionsfähigkeit von Systemen Verfahren für die Fehlzustandsart- und -auswirkungsanalyse (FMEA) (IEC 60812:2006); Deutsche Fassung EN 60812:2006," 2006.

Appendix A Enhanced Use Case Template

This appendix shows an empty use case template (UC Template) based on IEC 62559 [62] and enhanced with our proposed adjustments highlighted in green.

UC Template Section 1 – Description of the use case

UC Template Subsection 1.1 – Name of use case

Use case identification		
ID	**Area Domain(s) / Zone(s)**	**Name of use case**

UC Template Subsection 1.2 – Version management

Version management				
Version no.	**Date**	**Name of author(s)**	**Changes**	**Approval status**

UC Template Subsection 1.3 – Scope and objectives of use case

Scope and objectives of use case		
Scope	**ID**	**Name**
Objective(s)	**ID**	**Name**
Related business case(s)		

UC Template Subsection 1.4 – Narrative of use case

Narrative of use case
Short description
Complete description

UC Template Subsection 1.5 – Key performance indicators

Key performance indicators			
ID	**Name**	**Description**	**Reference to mentioned use case objectives**

UC Template Subsection 1.6 – Use case conditions

Use case conditions	
Assumptions	
ID	**Description**
Prerequisites	

UC Template Subsection 1.7 – Further information to the use case for classification / mapping

Classification Information
Relation to other use cases
Level of depth
Prioritization
Generic, regional or national relation
Nature of the use case
Further keywords for classification

UC Template Subsection 1.8 – General remarks

General remarks

UC Template Section 2 – Diagrams of use case

Diagram(s) of use case

UC Template Section 3 – Technical details

UC Template Subsection 3.1 – Actors

Actors			
Grouping		Group description	
Actor name	Actor type	Actor description	Further information specific to this use case

UC Template Subsection 3.2 – References

References						
No.	References type	Reference	Status	Impact on use case	Originator / organization	Link

UC Template Section 4 – Step by step analysis of use case

UC Template Subsection 4.1 – Overview of scenarios

Scenario conditions							
No.	Scenario name	Scenario description	Primary actor	Triggering event	Precondition	Postcondition	Objective (ID / Name)

UC Template Subsection 4.2 – Steps – Scenarios

Scenario										
Scenario name:										
Step No.	Event	Name of process/ activity	Description of process/ activity	Service	Information producer (actor)	Information receiver (actor)	Information exchanged (IDs)	Requirements R-ID	Assumption ID	Alternative scenario ID

UC Template Section 5 – Information Exchanged

Information exchanged			
Information exchanged ID	Name of information exchanged	Description of information exchanged	Requirements IDs

UC Template Section 6 – Requirements (optional)

Requirements (optional)			
Categories ID	Category name for requirements	Objective ID	Category description
Requirement ID	Requirement name	Objective ID	Requirement description

UC Template Section 7 – Common Terms and Definitions

Common terms and definitions	
Term	Definition

UC Template Section 8 – Custom information (optional)

Custom information (optional)		
Key	**Value**	**Refers to section**

Appendix B Classification of Hazardous Events

ISO 26262-3:2018 [5] introduces the following qualitative metrics to assess the different dimensions of a hazardous event: exposure, severity, and controllability.

Table 22 – Exposure classification ISO 26262-3:2018 [5, p. 8]

Class	**E0**	**E1**	**E2**	**E3**	**E4**
Description	Incredible	Very low probability	Low probability	Medium probability	High probability

Table 23 – Severity classification ISO 26262-3:2018 [5, p. 8]

Class	**S0**	**S1**	**S2**	**S3**
Description	No injuries	Light and moderate injuries	Severe and life-threatening injuries (survival probable)	Life-threatening injuries (survival uncertain), fatal injuries

Table 24 – Controllability classification ISO 26262-3:2018 [5, p. 9]

Class	**C0**	**C1**	**C2**	**C3**
Description	Controllable in general	Simply controllable	Normally controllable	Difficult to control or uncontrollable

Table 25 – ASIL determination ISO 26262-3:2018 [5, p. 10]

Severity class	**Exposure class**	**Controllability class**		
		C1	**C2**	**C3**
S1	**E1**	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	ASIL A
	E4	QM	ASIL A	ASIL B
S2	**E1**	QM	QM	QM
	E2	QM	QM	ASIL A
	E3	QM	ASIL A	ASIL B
	E4	ASIL A	ASIL B	ASIL C
S3	**E1**	QM	QM	ASIL A
	E2	QM	ASIL A	ASIL B
	E3	ASIL A	ASIL B	ASIL C
	E4	ASIL B	ASIL C	ASIL D

Appendix C Example Assurance Case

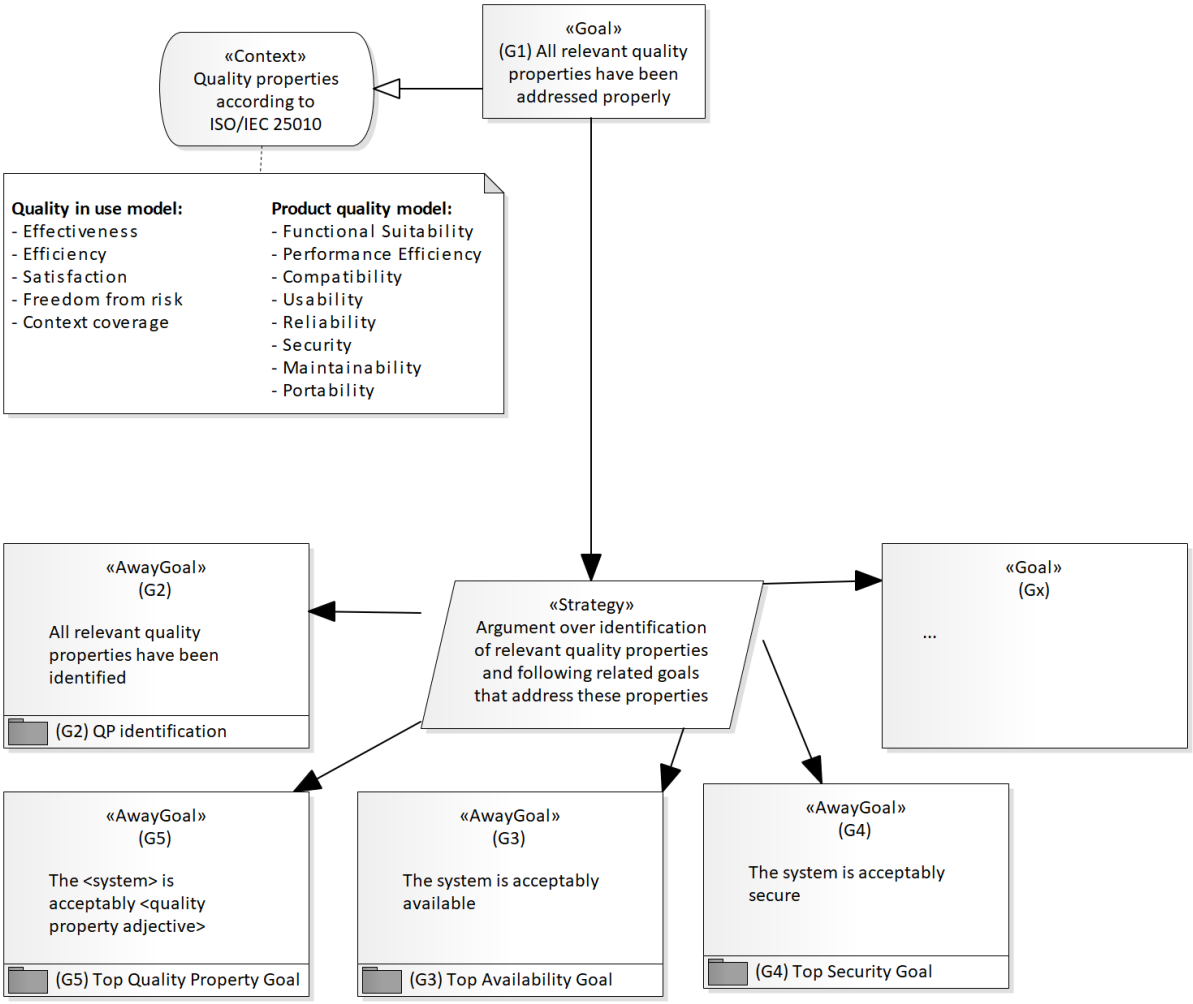


Figure 35 – Generic quality property assurance case

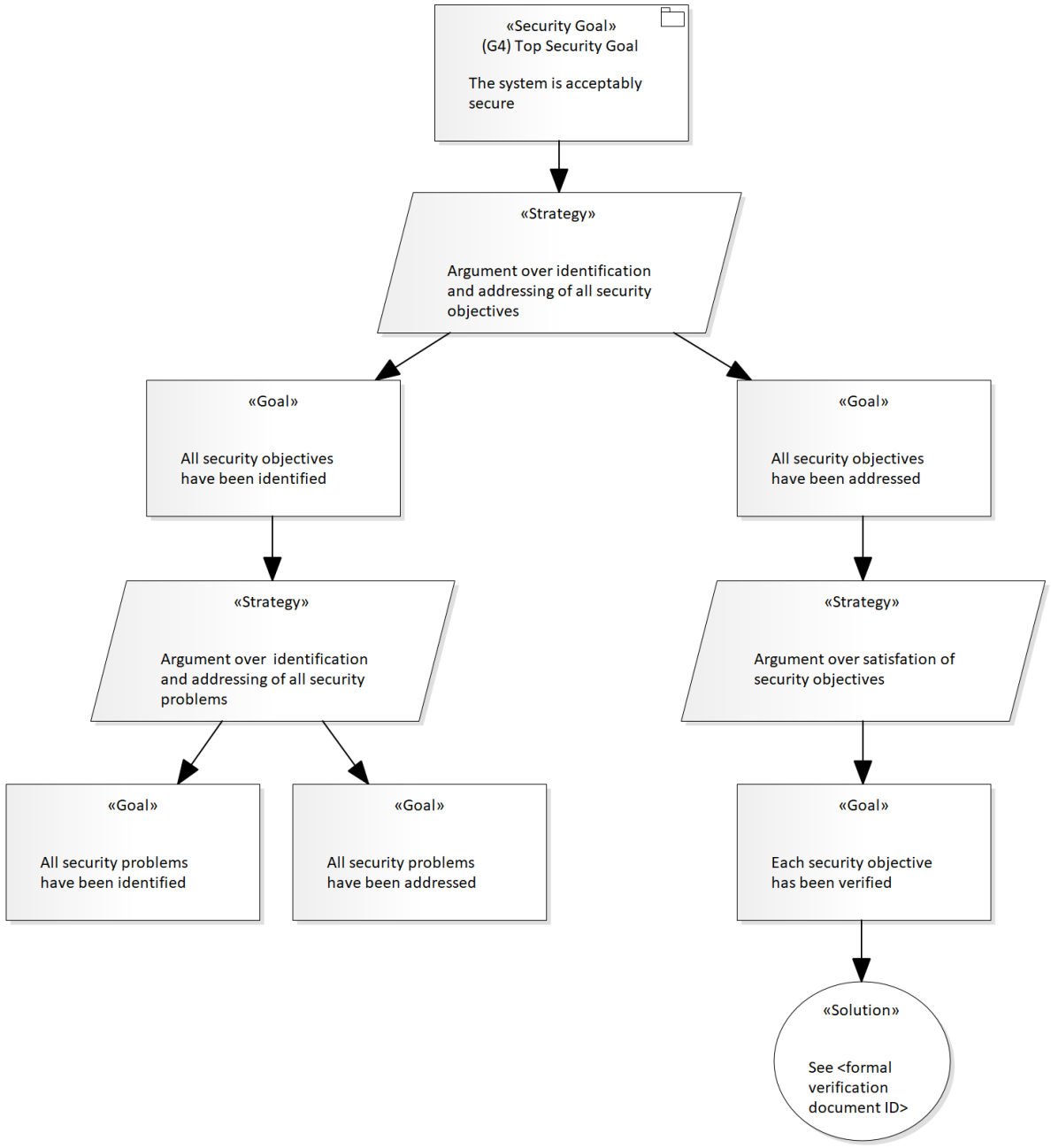


Figure 36 – Security Argument

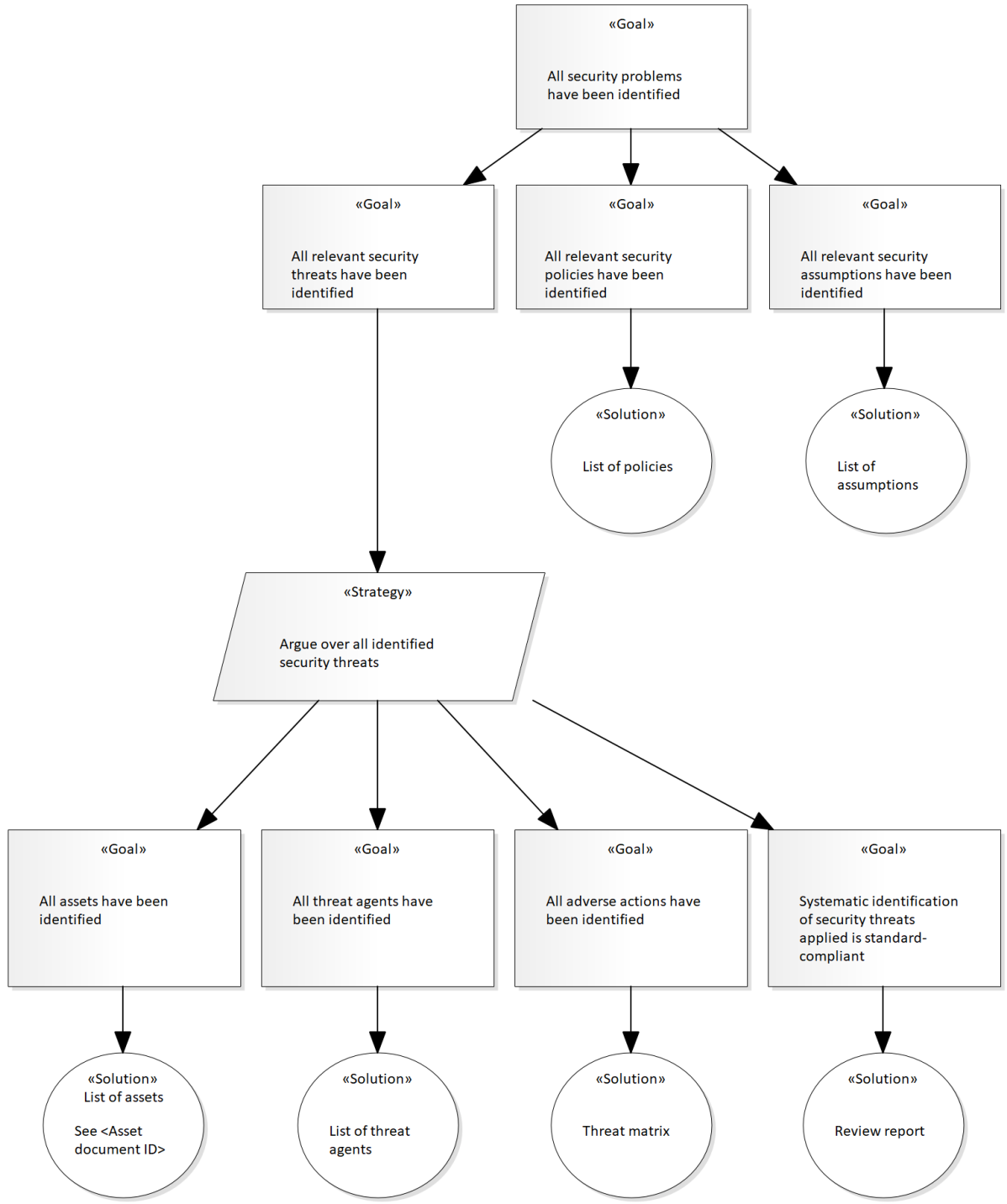


Figure 37 – Security Argument (continued) – security problem identification

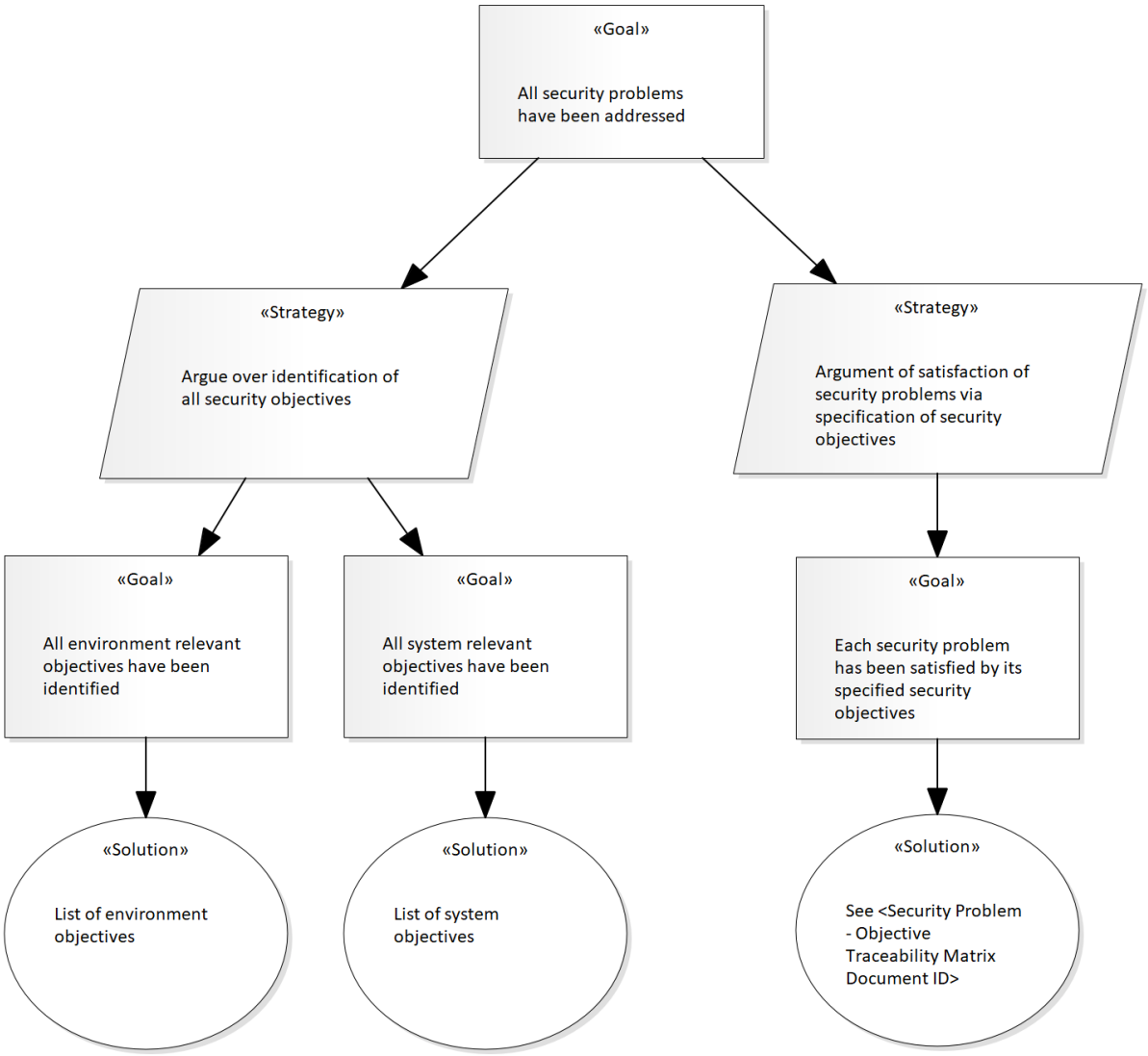


Figure 38 – Security Argument (continued) – security problem addressing

Document Information

Title:	Addressing Quality Properties in Use Case Descriptions
Date:	July 23, 2021
Report:	IESE-006.21/E
Status:	Final
Classification:	Public

Our future power grid is probably one of the most complex and most sophisticated critical infrastructures. Given the complexity of the smart grid and its safety and security challenges, requirements elicitation for smart grid solutions requires a systematic process to address these crucial non-functional system properties. The research described in this report was motivated by work carried out in the context of the "enera" project, a public research project funded by the German Federal Ministry for Economic Affairs and Energy (BMWi) within the funding program "Smart Energy Showcase – Digital Agenda for the Energy Transition" (SINTEG). Among the many project partners, the two research institutes Fraunhofer Institute for Experimental Software Engineering IESE and OFFIS – Institute for Information Technology conducted research on methods for systems and software engineering for complex systems and systems-of-systems. The outcome is a process for eliciting system-of-systems requirements, as well as an extension of the IEC 62559-2 use case template to address non-functional requirements. The activities in the use case elicitation process are accompanied and supported by an assurance case.

ISBN 978-3-8396-1724-3