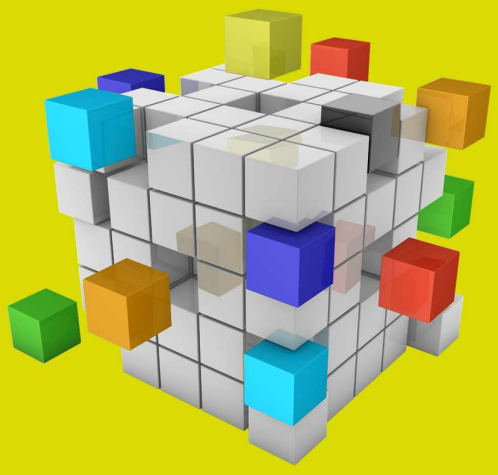
invenio



Whitepaper

# INTEGRATED SYSTEMS, SOFTWARE, AND HARDWARE ENGINEERING TO OVERCOME ENGINEERING SILOS

| VERSION | DATE | AUTHOR | CHANGES |
|---|---|---|---|
| 1.0 | 30.09.2024 | Enrico Seidel | Translation from 1.0 GE |
| | | | |
| | | | |

## CONTENT

## Summary

The growing complexity of modern systems requires a shift from isolated engineering silos to an integrative approach that encompasses all engineering disciplines. The invenio Systems Engineering Framework (iSEF) by invenio AG bridges the gaps between systems, software, and hardware engineering with a new development tool, effectively implementing a newly formulated architecture-centered model-based systems engineering (ACMBSE). By formally and systematically linking all model views and abstraction levels within the architecture, the iSEF promotes interdisciplinary collaboration and ensures a coherent development process. Additionally, a practice-oriented ACMBSE engineering training program is currently being developed to support the resolution of engineering silos.

# 1 INTRODUCTION

The convergence of advanced technologies in the automotive, aerospace, and other high-tech industries has led to unprecedented system complexity recently [3]. Traditional engineering approaches, often characterized by document-oriented, isolated work within disciplines (silos), have struggled to manage this complexity effectively. The seamless integration of Systems Engineering (SE), Software Engineering (SWE), and Hardware Engineering (HWE) is therefore more important than ever.

In response to this challenge, invenio Systems Engineering GmbH introduces a framework (iSEF). By linking model views across all system abstraction levels, this framework highlights the strengths of architecture-centered model-based systems engineering (ACMBSE) and enables comprehensive and coherent architecture descriptions [5].

This white paper provides an overview of the fundamental principles of iSEF, its practical application in the automotive domain as an example [4], and the benefits of breaking down silos to achieve synergies between engineering domains.

# 2 CHALLENGES IN OVERCOMING ENGINEERING SILOS

Historically, systems, software, and hardware engineering developed independently, each with its own methods, tools, and cultures, particularly concerning requirements and architecture [6,7]. This separation often led to misinterpretations, redundant documentation, and inefficiencies. In complex projects with significant dependencies between systems, software, and hardware, the lack of integration capability becomes problematic. Text-based requirements engineering often remained the only link for communication between these disciplines [4]. Therefore, traditional document-based approaches are no longer sufficient. An architecture-centered, model-based approach (ACMBSE) based on MBSE principles, supported by a robust framework like iSEF, is essential to address these challenges. The following will outline some key aspects of these challenges and their possible solutions [3].

## 2.1 Lack of Proper Technical Communication

One of the greatest challenges developing in engineering silos is the lack of effective communication and coordination between engineering disciplines, which creates barriers to collaboration.

Different methods, tools, languages, and notations lead to misunderstandings and miscommunication, complicating the alignment of technical content between teams. This results in fragmented knowledge and disconnected development artifacts, which can significantly hinder the efficiency and effectiveness of engineering projects.

Therefore, a shared technical language is crucial. The graphical notation UML/SysML [8], when properly learned and applied, greatly facilitates communication, as a systematically created visual representation can significantly better illustrate the relationships of system behavior or structure than textual requirements [9]. This also significantly improves communication efficiency, as engineers no longer need to mentally "model" their own view of the system's structure and behavior to comprehensively understand the behavior of a system element [10].

### 2.2 Redundant Efforts and Resulting Inefficiencies

Isolated engineering teams often generate duplicated efforts and redundant work, as they tend to address the same problem independently across different projects. This redundancy wastes resources, increases development costs, and delays project timelines, particularly in large development projects.

The solution is again an architecture-centered model-based approach (ACMBSE) [3], where system components are decomposed and structured according to model decomposition principles. By applying procedures for creating standardized interfaces and services within an interface database, reusable model components can be established, significantly enhancing the efficiency of platform projects [11].

### 2.3 Inconsistent Data and Models

The most important principle, which is quickly forgotten in the hustle and bustle of projects, is the Single Source of Truth (SSOT) [12]. Without a unified work environment and methodology for creating the necessary engineering artifacts, different teams develop incompatible models, complicating the integration and validation processes. This inconsistency often leads to malfunctions due to faulty designs [6,7].

A model-based and systematically maintained source of information significantly enhances efficiency and reduces the effort required for data collection, as long as it is ensured that all team members always have access to up-to-date data. This ultimately minimizes the risk of project errors. Furthermore, It is crucial that all disciplines work with a methodology and unified processes coordinated between the disciplines [6]. An engineering framework that prescribes this way of working is therefore becoming increasingly important for implementing structured work with SSOT data.

### 2.4 Delayed Problem Identification

In isolated development teams and engineering environments, cross-disciplinary problems often are undetected until late in the development cycle. Hardware limitations that impair software performance might only be discovered during the integration phase, making their resolution more costly. Engineering silos further hinder the necessary visibility and communication [13].

A unified cross-disciplinary method in model creation can provide a remedy. This requires that the software and hardware architecture models be derived from a common system model. The best results are achieved when all disciplines work within the same model and follow the same set of rules. It is particularly important to precisely define the Hardware-Software Interface (HSI) [5]. Additionally, executable functional models at both the system and discipline levels enable early evaluation of system behavior. However, this requires disciplined project management to allocate time for verifying the architecture before commencing actual development.

### 2.5 Limited Reusability

A development team that lacks foresight and only thinks within the context of its own project limits the reuse of created components and designs. Teams often develop custom solutions for successive projects repeatedly, rather than using existing components, improving them, and making them available again to the development pool. This reduces overall efficiency, as teams are unable to benefit from previously solved problems and tested solutions.

A solution is to establish an organizational structure that focuses on functional development and makes these solutions available to platform teams. These teams then develop technical solutions that can be reused in various future customer projects. However, this requires management foresight, including making initial investments that will pay off through the use of validated and quickly integrable artifacts. The prerequisites for this include appropriate integration reference models and integration specifications. iSEF supports this integration with pre-built model libraries.

### 2.6 Hindrance of Innovation Due to Lack of Competence

Innovation thrives in environments where diverse perspectives and ideas can mutually influence one another. However, based on the author's experience, many systems engineers lack sufficient understanding of the challenges faced in hardware and software disciplines and create highly abstracted system models that offer little benefit to discipline engineers. Furthermore, customer requirements are often inadequately analyzed, leaving the true requirements unclear and shifting problems down to the discipline level.

Good system design requires a deep technical understanding of the system to be developed and the resulting system architecture, along with the subsequent work of hardware and software engineers. This knowledge should be imparted through interdisciplinary and technically sound training that includes essential foundations in both software and hardware disciplines. Existing programs such as CSE [14], INCOSE certifications [15], or training for "Embedded System Architect" [16] offer the right approaches but lack the practical aspect of creating a system architecture that can be further utilized across disciplines.

At invenio Systems Engineering GmbH, we are currently developing a training program based on the ACMBSE approach. iSEF serves as the technical foundation for modeling in this program. The training will consist of four levels and will be similar to the OCSMP program by OMG [17].

- Applied System Modeling with iSefML – Fundamentals and ACMBSE Methodology
- Applied System Modeling with iSEF – Formal System Analysis & System Design
- Applied System Modeling with iSEF – Behavioral Modeling of Complex Systems
- Applied System Modeling with iSEF – Development of Domain-Specific Model Libraries

# 3 THE INVENIO SYSTEMS ENGINEERING FRAMEWORK

The invenio Systems Engineering Framework (iSEF) provides essential solutions for overcoming engineering silos by introducing a unified technical language and methodology based on UML/SysML [8]. Currently implemented in IBM Rhapsody, it enables user-guided, structured storage of model components, resulting in a coherent data model. Supported by customizable model checks, iSEF ensures that all disciplines work consistently. A shared system model also promotes the early detection and resolution of potential issues, facilitating system integration and validation. Furthermore, iSEF supports the efficient development of reusable components through reference models and specifications.

## 3.1 The Three Dimensions of the iSEF Architecture Description

As system complexity increases [1], systems need to be divided and organized into smaller units or subsystems. Each subsystem has a specific function, allowing architects and designers to focus on the details of that particular part. To facilitate this, iSEF introduces three dimensions of system architecture based on the work in [4,5]: the different views on the system (*Viewpoints*), the abstraction of a system (*System Abstraction Levels*), and the decomposition granularity (*Hierarchy Depth*). These three orthogonal dimensions enable a structured approach in which all architecture elements are formally stored in a virtual 3D space, as shown in Figure 1. To facilitate understanding, only specific sections (*viewpoints*) are shown by the modeler, with all elements transposed onto a 2D space according to UML/SysML [8] (see Figure 1).

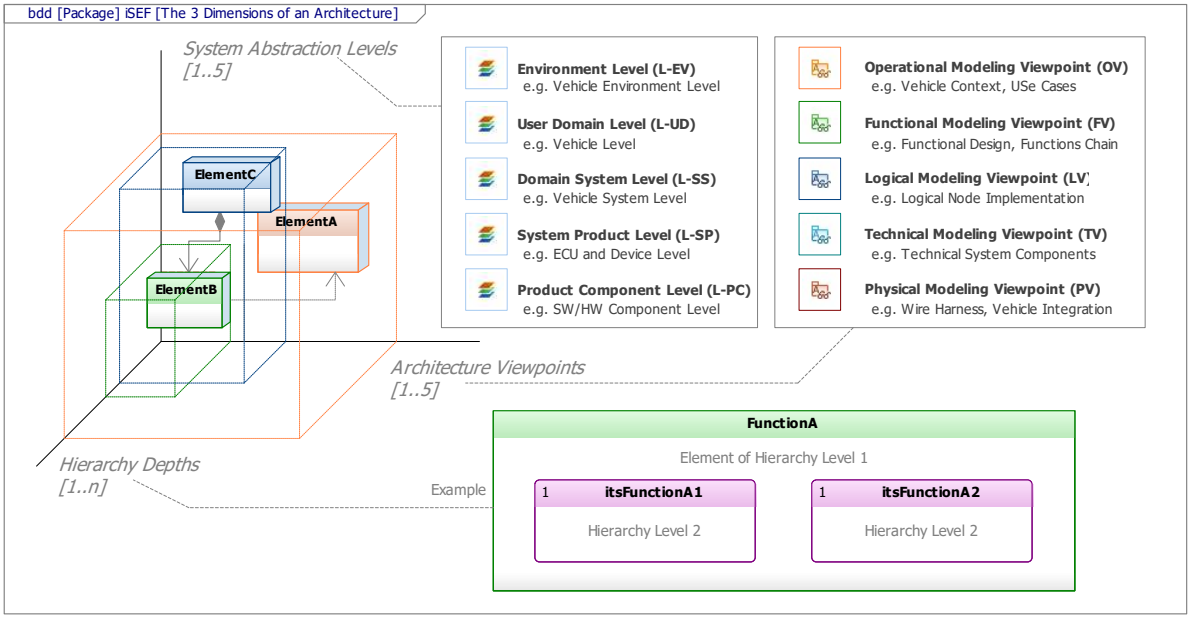



Figure 1: The Three Dimensions of System Architecture Description

### 3.1.1 The Viewpoints of iSEF

As part of the development of this framework [2], five architecture-relevant *viewpoints* were identified according to Figure 1 to analyze, define, and create a system with a strong focus on reusability: the *Operational Viewpoint*, the *Functional Viewpoint*, the *Logical Viewpoint*, the *Technical Viewpoint*, and the *Physical Viewpoint* [5]. The iSEF rule set defines which type of element may be created on which viewpoint and corresponding abstraction level. The Rhapsody tool, with the iSEF profile, supports correct modeling through stringent user guidance.

### 3.1.2 Linking Elements Between Viewpoints

To ensure a coherent architecture description across all viewpoints, iSEF introduces extended SysML notations and semantics. Elements from different viewpoints are linked together [5]: functional elements are integrated into logical ones, which are then composed into technical elements, and finally into physical structural elements. This linking is achieved through interfaces (ports), as shown in Figure 2, which connect across the viewpoints [5]. This formal interplay of element composition and interface connections ultimately allows for formal modeling with automated consistency checks using an interactive model checker during the modeling process.

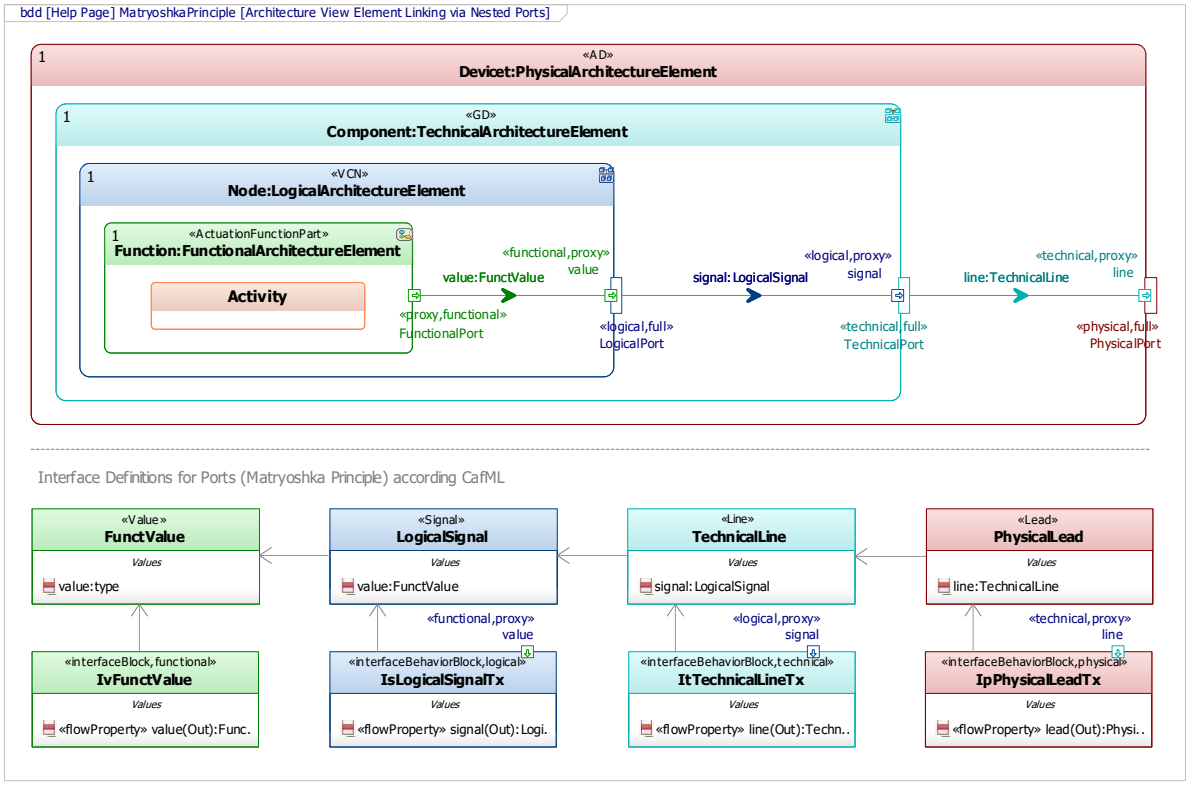



Figure 2: Linking of Elements and Interfaces Across Sequential Viewpoints

Thanks to this method, formal consistency of interfaces is achieved through nested port modeling, where *Proxy Ports* are embedded in full ports via the interface definition of an *Interface Behavior Block* [5]. Functional values are encapsulated by logical signals and transmitted over technical interfaces, which are bound to physical interfaces. Functional, logical, technical, and physical ports reflect the interfaces to their respective preceding viewpoints. This principle is also known as the “Matryoshka” principle.

The methodology of embedding elements from one viewpoint into the subsequent one provides the most robust formal solution for linking all viewpoints and is the central key to architecture modeling in iSEF. Particularly for systems requiring high-functional safety up to ASIL C/D, this method ensures conformity with the requirements of ISO 26262.

Another advantage resulting from the ACMBSE method is the flexible composition of reusable structures across viewpoints, as the shown interfaces of an *Interface Behavior Block* [5] also perform a transformation role. This allows a component's logical interface to appear constant externally, even if an internal value transformation occurs within the interface, for example, when a functional block with a similar but not identical interface is embedded. Modeling tools implementing this method can apply these principles to automate notations and semantic verifications, achieving machine-based model verifiability.

### 3.1.3 The Abstraction Levels of iSEF

The functional and technical scope of a system, as well as the depth of analysis into the system, is defined by *System Abstraction Levels* [4], which represent an artificial demarcation rather than an inherent property of the system. These levels serve as a classification for any possible system. On the other hand, a system abstraction level may pertain to the process of generalizing the physical, spatial, or temporal details of a system, or the abstraction may focus on a user-perceived model or representation of physical information.

In iSEF, this artificial demarcation is divided into five *System Abstraction Levels*: the *Environment Abstraction Level* (L-EV), the *User Domain Abstraction Level* (L-UD), the *Domain System Abstraction Level* (L-DS), the *System Product Abstraction Level* (L-SP), and the *Product Component Abstraction Level* (L-PC). A detailed description can be found in [4], and an example representation of the scope of the abstraction levels is provided in Figure 1. Any system and its subsystems can be classified into one of the abstraction levels [2]. When decomposing a system, a subsystem can immediately be assigned to a lower abstraction level if no further hierarchy is applied. The development always begins at the level of the system context, the System of Interest.

System abstraction levels offer several advantages: they enable the derivation of new system contexts from subsystems with clear interactions between system elements such as functions, nodes, and components, and they prevent "level-hopping" between elements of different abstraction levels. Additionally, they allow for the implementation of machine-verifiable decomposition rules for architecture elements at the respective abstraction levels, based on mandatory decomposition models. This ensures that teams adhering to these rules consistently create compatible system structures.

Supported by standardized interfaces from the iSEF libraries, architectures are thus created that enhance communication between development teams and provide reusable system components.

### 3.1.4 Hierarchy Depth (Hierarchical Model Depth)

When a system is analyzed on a specific abstraction level and modeled across all viewpoints, this is referred to as a *Full Architecture Step*. Within this step, system, and functional components may exhibit different levels of granularity across the viewpoints.

The hierarchical model depth [4] thus defines the granularity of the decomposition of a system within a specific viewpoint's system abstraction level. To keep the decomposition straightforward, a preferred hierarchical depth of one is recommended. However, the hierarchical depth between viewpoints within a system abstraction level does not have to be constant, as granularity can vary depending on the decomposition scheme.

To ensure a standardized system architecture, iSEF offers a series of reference models that guide architects in designing system components across the viewpoints in such a way that a reusable architecture step can be easily modeled. An example representation of the hierarchical model depth is provided in Figure 1.

### 3.2 Methodical Modeling Steps for System Design

Building on the three dimensions of architecture description, as outlined in [18], system modeling with ACMBSE and iSEF is made possible through eight fundamental steps. These include feature definition, requirement derivation, system context definition, behavior analysis, functional design, logical component development, technical system development, and the realization of the physical architecture.

When the method presented in [18] is correctly applied, it results in an architecture description that can be coherently represented across all abstraction levels and viewpoints, extending to the discipline-specific architectures of hardware and software.

Naturally, it is not always practical to model the entire 3D space of elements. Rather, the right balance is needed [2], focusing on essential paths within the modeling space. The tailoring schemes included in iSEF assist the modeler in identifying the necessary path and representing all critical elements effectively.

## 4 EXCELLENCE THROUGH APPLIED SYSTEMS ENGINEERING

An essential aspect for the proper development of a system architecture is not only the provided framework and model libraries, but also the appropriate training of engineers to use these tools correctly. This approach helps to break down engineering silos and promotes a coherent way of working. A training program, as suggested below, can teach the relevant methodology. It should provide a solid foundation in the ACMBSE approach, which is then trained in practical exercises using modeling tools. The key elements of this training should cover the following content:

- Fundamentals of architecture-centered systems engineering, focusing on the principles of ACMBSE as outlined in this paper
- Specializations of the extended iSEF modeling language based on SysML/SysML V2 and UML, covering all engineering disciplines
- Teaching and applying all decomposition models of the corresponding model views and integrating them into a coherent model
- Application of the ACMBSE-trained methodology and notation in the implementation of given architectures, specifically in EE development
- Detailed modeling of simulation-capable functional model structures from scratch using existing iSEF functional blocks
- Modeling of technical and physical system architectures that can be seamlessly transferred into disciplinary architectures

- Modeling of eco-system architectures and cloud-based systems to implement currently existing systems
- Training and fostering interdisciplinary collaboration between software and hardware disciplines and the overarching systems engineering discipline
- Creation of an end-to-end model from system concept to realization within disciplines during the course, through group work and self-study

A central role in this process is establishing the ability for all system and discipline engineers to collaborate, understand, and read the modeled content of other disciplines. Moreover, engineers should create their own models in such a way that they are comprehensible and usable for the respective other disciplines, conveyed in the same language.

## 5 RESEARCH NEEDS AND OUTLOOK

The invenio Systems Engineering Framework (iSEF) addresses the challenges of overcoming engineering silos by promoting an integrated approach that links different perspectives and strengthens collaboration between the disciplines of systems engineering, software engineering, and hardware engineering. Through the use of abstraction levels and model views, coherent data models are ensured, effectively breaking down silos within engineering. This not only improves communication and coordination between engineering teams but also enhances the ability to identify and resolve cross-disciplinary issues early on, ensures data consistency, and promotes the reuse of components and artifacts—ultimately driving innovation and efficiency in projects.

An efficient application of the iSEF requires not only adherence to its rules, but also an initial investment in creating out-of-the-box model libraries. In the coming months, further development work is required, particularly in domain-specific signal libraries and basic functional building blocks. Once the foundational libraries are in place, a “Model Store” is envisioned to provide users with an extensive ecosystem of standardized and compatible model libraries, developed by and with the systems engineering community.

Another important aspect is the future use of generative artificial intelligence (AI) in the analysis and modeling of systems. Initial research is currently being conducted to explore the use of AI agents. These agents can assist in evaluating models for plausibility and take over repetitive tasks in the creation of interfaces and standardized system components. The deployment of AI enables efficient creation and adaptation of these models, resulting in significant quality and time benefits for system development.

To promote broader application of this framework, additional training programs should be developed in collaboration with universities, and the methodology should be didactically systematized. Practical training plays a crucial role in ensuring that engineers acquire the necessary skills and knowledge for efficient use of the iSEF.

# 6 REFERENCES


[1] T. Weilkiens, Systems Engineering mit SysML/UML: Anforderungen, Analyse, Architektur. Mit einem Geleitwort von Richard Mark Soley. Heidelberg, Germany: dpunkt.verlag, 2014.

[2] T. Weilkiens, SYSMOD - The Systems Modeling Toolbox - Pragmatic MBSE with SysML. Morrisville, NC: Lulu.com, 2016.

[3] M. D. Brooks and T. M. Wheeler, "Experiences in Applying Architecture-Centric Model-Based System Engineering to Large-Scale, Distributed, Real-Time Systems," in Proceedings of the 2007 IEEE/ACM International Conference on Software Engineering (ICSE 2007), 2007.

[4] E. Seidel and M. Forlingieri, "Moving towards Server-Zone Architecture with MBSE at Continental," in INCOSE International Symposium, Hawaii, USA, Jul. 2023. [Online]. Available: https://www.incose.org/symposium2023.

[5] E. Seidel and M. Forlingieri, "Architecture-Centric Model-Based Development At Continental," in Proceedings of the AOSEC International Symposium, Bangalore, India, Oct. 2023. [Online]. Available: https://www.aosec2023.org.

[6] "Systems and software engineering — Architecture description," ISO/IEC/IEEE 42010:2011, 2011.

[8] "OMG Systems Modeling Language Version 1.6," OMG SysML, Dec. 2019. [Online]. Available: https://www.omg.org/spec/SysML/1.6.

[9] "Study of Modern Modeling Techniques for Model-Based Systems Engineering Methodologies," IJERT, Feb. 2023. [Online]. Available: https://www.ijert.org/study-of-modern-modeling-techniques-for-model-based-systems-engineering-methodologies.

[10] "Mastering State Diagrams in UML: A Comprehensive Guide," Visual Paradigm Guides, 2023. [Online]. Available: https://guides.visual-paradigm.com/mastering-state-diagrams-in-uml-a-comprehensive-guide/.

[11] "Engineering Silos," Heykona, 2023. [Online]. Available: https://www.heykona.com/blog/engineering-silos.

[12] "Reimagining Engineering to Deliver More Projects More Efficiently," McKinsey & Company, 2023. [Online]. Available: https://www.mckinsey.com/capabilities/operations/our-insights/reimagining-engineering-to-deliver-more-projects-more-efficiently.

[13] A. de Waal, M. Weaver, T. Day, and B. van der Heijden, "Silo-Busting: Overcoming the Greatest Threat to Organizational Performance," Sustainability, vol. 11, no. 23, Article 6860, Dec. 2019. [Online]. Available: https://doi.org/10.3390/su11236860.

[14] "Systems Engineering," OOSE. [Online]. Available: https://www.oose.de/unsere-themen/systems-engineering.

[15] "Certification," INCOSE. [Online]. Available: https://www.incose.org/certification. [Accessed: June 7, 2024].

[16] G. Muller, L. van Veen, and J. van den Aker, "Systems Engineering Education: From Learning Program to Business Value," MDPI Systems, 2023.

[17] "OCSMP Certification," OMG. [Online]. Available: https://www.omg.org/certification. [Accessed: June 7, 2024].

[18] E. Seidel "Architecture-Centric-Model-Based-Systems-Engineering-ACMBSE" invenio. [Online]. Available:https://www.invenio.net/systems-engineering/wp-content/uploads/sites/7/Architecture-Centric-Model-Based-Systems-Engineering-ACMBSE_DE.pdf. [Accessed: October 30, 2024].