

Article

Towards a Unified Engineering Approach for Variability and Modular Architecture Management in Automotive Systems

Fabian Goihl ^{1,†}, Yannick Lindebauer ^{2,*,†} , Richard von Eisebeck ^{3,†} , Jivka Ovtcharova ⁴ and Thomas Vietor ² 

¹ Dr. Ing. h.c. F. Porsche AG, Porschestraße 911, 71287 Weissach, Germany; fabian.goihl1@porsche.de

² Institute for Engineering Design, Technische Universität Braunschweig, Hermann-Blenk-Str. 42, 38108 Braunschweig, Germany; t.vietor@tu-braunschweig.de

³ Volkswagen AG, 38436 Wolfsburg, Germany; richard.von.eisebeck@volkswagen.de

⁴ Institute for Information Management in Engineering, Karlsruhe Institute of Technology, Kaiserstraße 12, 76131 Karlsruhe, Germany; jivka.ovtcharova@kit.edu

* Correspondence: yannick.lindebauer@tu-braunschweig.de

† These authors contributed equally to this work.

Abstract

Automotive systems are experiencing a rapid increase in complexity driven by the transition towards software-defined vehicles, autonomous functionalities and increasingly interconnected E/E architectures. This transformation intensifies variability across hardware and software domains and challenges established engineering approaches. Traditional modular product development (MPD) provides structural mechanisms to manage hardware complexity, while systems and software product line engineering (SPLE) offers methods for managing software variability. However, these paradigms are typically applied in isolation and lack an integrated methodology capable of addressing cross-domain variability and architectural synchronization in automotive systems. This paper investigates how SPLE and MPD can be systematically integrated to manage variability and architectural complexity in automotive systems. Following a design-oriented research approach, industry requirements are derived from an automotive case study at an OEM. Existing SPLE and modularization approaches are analyzed against these requirements, revealing gaps in cross-domain traceability, synchronization mechanisms, and lifecycle coordination. Based on this analysis, we propose an integrated methodology that combines variability modeling principles from SPLE with architectural modularization concepts. The approach enables management of module structures, supporting system-level consistency in automotive environments. The main contribution is a model-based-integration framework that bridges variability management and modular architecture design to address increasing system complexity in the automotive industry.

Keywords: software product line engineering; modular product development; model-based systems engineering



Academic Editor: Tugba Karabiyik

Received: 12 June 2026

Revised: 3 July 2026

Accepted: 20 July 2026

Published: 27 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

The automotive industry is currently undergoing a fundamental transformation toward the paradigm of the software-defined vehicle (SDV). In this context, value creation is increasingly shifting from predominantly mechanical components toward software-intensive and mechatronic systems [1,2]. Functions are no longer primarily defined by hardware characteristics, but by software capabilities that can be continuously updated, extended, and configured over the vehicle lifecycle [3]. This shift significantly increases

system complexity and fundamentally changes the requirements for automotive product development processes.

Traditionally, automotive development has been largely organized around platform-based approaches [4]. Major original equipment manufacturers (OEMs) such as Volkswagen with the Modular Transverse Matrix (MQB) or the Modular Electric Drive Matrix (MEB), and General Motors with platforms such as Ultium [5], structure their product portfolios around shared hardware architectures. These platforms primarily address the mechanical and structural aspects of vehicles and have proven effective in managing hardware reuse and economies of scale. However, they are less suited to capture the growing importance and dynamics of software-driven functionality. In parallel to mechanical platforms, the electrical/electronic (E/E) architecture has evolved as a separate layer, defining the structure of control units, communication networks, and software deployment. While this architecture provides the backbone for software execution, it does not fully address how software behavior is adapted to specific vehicle configurations. A crucial third dimension therefore emerges in modern automotive systems: application data. Often realized as calibration parameters or configuration data, they enable the adaptation of generic software functions to the specific characteristics of a concrete vehicle instance and its variants. Through application data, identical software components can be configured to meet vehicle variants. As a result, application data plays a central role in bridging the gap between hardware platforms, software architectures, and the variability inherent in automotive product lines.

The result is an increasingly complex combinatorial interplay between platform structures, E/E architectures, and application data, which, in the context of SDVs, continues to grow in both scale and complexity.

Against this backdrop, modular product development (MPD) represents a further step beyond traditional platform strategies. Its objective is to enable the development of self-contained modules that can be selected, combined, and reused across different projects in a flexible, “shelf-like” manner. In practice, however, this vision has not yet been fully realized. Current platform-based development still requires substantial adaptation and integration effort for each new product. To enable true MPD, several fundamental questions must be addressed. These include how modules are defined and partitioned, how they are developed, and, most critically for this work, how they are systematically managed and controlled.

An established answer to this last question is systems and software product line engineering (SPLE), originating from software engineering, which provides mechanisms to manage variability and to enable systematic reuse of assets across different variants [6]. While SPLE offers a strong foundation for variability management, its application to MPD in the automotive domain remains insufficiently explored. In particular, it is unclear how modules can be effectively controlled using SPLE principles, which information is required to achieve this, and how such an approach can be structured in a complex, cross-domain environment.

This paper addresses this gap by investigating how SPLE can be extended and integrated with MPD to enable systematic module management in automotive systems. The main contributions are therefore:

- Identification and systematic description of the requirements of an automotive OEM for a methodology to manage modularization in complex systems, e.g., mechatronic systems.
- Proposal of a model-based integration framework for the coordinated management of modules, including its proof of concept in an industrial context and an initial evaluation.

The remainder of this paper is structured as follows. Section 2 introduces the necessary background on SPLE and MPD. Section 3 reviews related literature and positions the contribution within the existing body of research. In Section 4, the research approach is described, followed by Section 5, which presents an industrial application analysis conducted at Volkswagen. Based on this analysis, specific requirements are derived and subsequently compared with existing, general approaches in SPLE and MPD. Section 6 introduces the proposed methodology, which integrates both research streams into a unified approach. Section 7 shows an initial proof of concept of the methodology through an automotive example. The subsequent evaluation of the proposed approach is presented in Section 8. Section 9 discusses the threats to validity, while Section 10 concludes the paper.

2. Background

This chapter introduces the two existing research streams that are to be integrated into a new methodology for automotive systems. Section 2.1 first presents SPLE, followed by Section 2.2, which outlines the fundamental concepts of MPD.

2.1. Systems and Software Product Line Engineering (SPLE)

In ISO/IEC 26550:2015 [7], the objective of SPLE is defined as the creation, utilization, and management of a common platform. This platform enables the systematic development of a family of products, such as software systems or system architectures, while reducing costs, time-to-market, and at the same time ensuring a high level of quality in the resulting artifacts [6–10].

To realize such a platform, various variability frameworks have been proposed. A well-established approach is the two-dimensional SPLE framework described in the literature [9,11] as shown in Figure 1. This framework is structured along two orthogonal axes: domain engineering and application engineering on the one hand, and problem space and solution space on the other. Together, these dimensions form a conceptual model that serves as the foundation for describing and managing variability. The framework captures technical variability and systematically relates it to software assets. Based on specific variant configurations, the appropriate assets are derived, ensuring that each variant is equipped with artifacts precisely tailored to its configuration [10].

A clear understanding of these dimensions is essential, as their distinction is not merely terminological but conceptual. The separation between domain and application engineering reflects a process-oriented perspective, in which domain engineering encompasses activities concerned with identifying, modeling, and managing commonalities and variability, whereas application engineering focuses on the reuse and instantiation of these artifacts in concrete products [7,10].

The distinction between problem space and solution space complements this view by differentiating between types of artifacts. The solution space contains tangible development artifacts, such as software components or system elements. In contrast, the problem space represents variability itself, i.e., the set of features or characteristics that are used to control and configure these artifacts [9,11]. Maintaining a strict separation between these spaces is crucial, as it enables a consistent and traceable representation of variability across the development process.

Within the quadrant defined by problem space and domain engineering, variability itself is described and structured. This includes the identification and modeling of features and their relationships, forming the conceptual basis for subsequent derivations. In contrast, the quadrant combining solution space and domain engineering focuses on the definition of concrete artifacts within the development process, such as reusable components, architectures, or assets that realize the modeled variability. Moving to the application perspective,

the problem space in application engineering is concerned with the specification of concrete variants. These variants are defined through the selection and configuration of features. Finally, in the quadrant of solution space and application engineering, these configured variants are materialized into concrete product instances. This includes the generation of variant-specific artifacts, such as configuration files or deployable software assets, which directly correspond to the selected feature combinations.

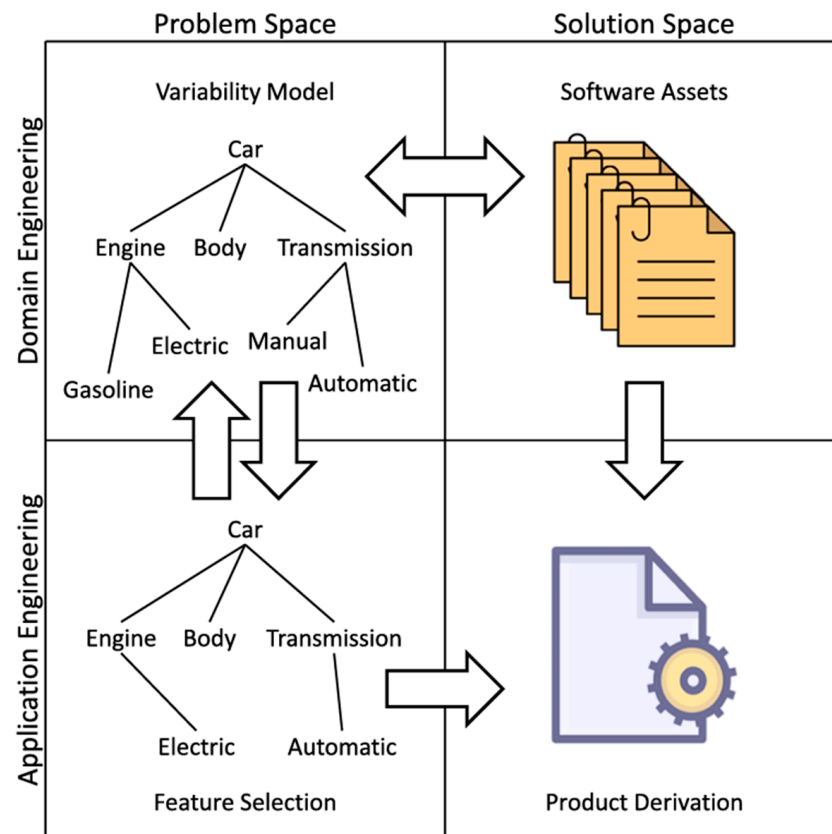


Figure 1. Variability management framework [9,11,12].

Variability in the problem space can be modeled using different approaches and languages, e.g., the orthogonal variability model (OVM) [6] and feature-based modeling [13,14]. Feature modeling, in particular, has become one of the most widely adopted approaches [15]. In this paper, feature models are therefore used as the primary means for representing variability. According to ISO/IEC 26580:2021 [16] a feature is defined as a characteristic of a product that distinguishes it from other products within a product line.

Feature modeling was originally introduced more than 35 years ago as part of the feature-oriented domain analysis (FODA) method [17]. Typically, feature models, shown in Figure 2, are organized as hierarchical, tree-like structures, in which features are arranged to represent all possible configurations. Relationships between features are defined through rules, commonly expressed using Boolean operators, which enable the derivation of valid configurations [13]. In addition, features are assigned to specific group types, such as mandatory, optional, or alternative. These group types determine whether a feature must be selected, may be selected, or whether exactly one feature from a group must be chosen. Furthermore, feature models incorporate cross-tree constraints, which allow dependencies between features beyond the hierarchical structure to be expressed.

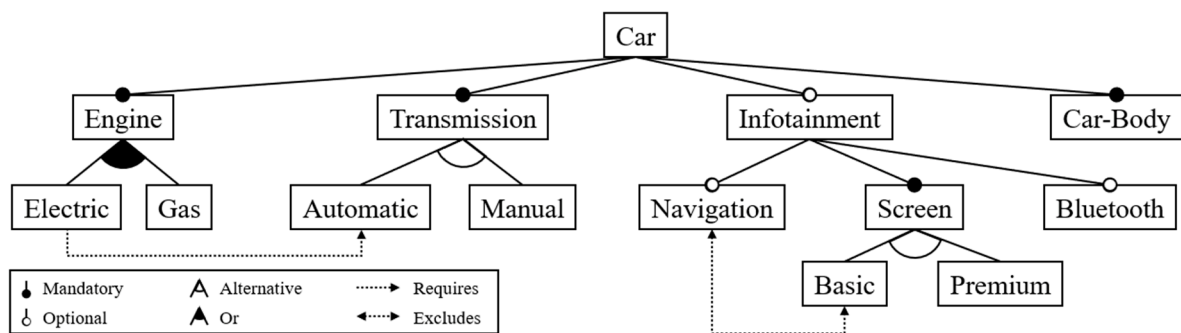


Figure 2. Exemplary feature model of a car [18].

2.2. Modular Product Development (MPD)

Product modularity is conceptualized in diverse ways within the literature, reflecting its impact across all phases of the product lifecycle and the differing objectives associated with these phases. Existing definitions can be broadly structured along several perspectives, including structural (coupling, decoupling, and interfaces), technical, functional, process and organizational, and product-strategic viewpoints [19,20].

While the structural perspective emphasizes strong intra-module coupling and weak inter-module coupling as a defining characteristic [19–21], the technical perspective emphasizes the physical and structural decomposition of products into manageable elements based on engineering relations and dependencies [22]. The functional perspective interprets modularity from the standpoint of functional decomposition. Modules are defined based on their responsibility for fulfilling specific product functions or sets of functions [22–24]. Additional perspectives include the process and organizational perspective, which considers the implications of modular product structures for development and production processes [25], as well as the product-strategic perspective, which emphasizes the role of modularity in enabling product family planning and market-oriented product differentiation [19].

Across these perspectives, a common understanding emerges in which modularity is grounded in the deliberate concentration of dependencies within modules and the reduction in dependencies between them. However, this principle alone is insufficient to fully describe modular product architectures. The decoupling of modules constitutes a necessary condition, as it enables independent development, modification, and substitution. To more comprehensively characterize modularity, additional properties must be considered, including communal use, combinability, interface standardization, and functional binding. An overview of the characteristics of modularity is shown in Figure 3.

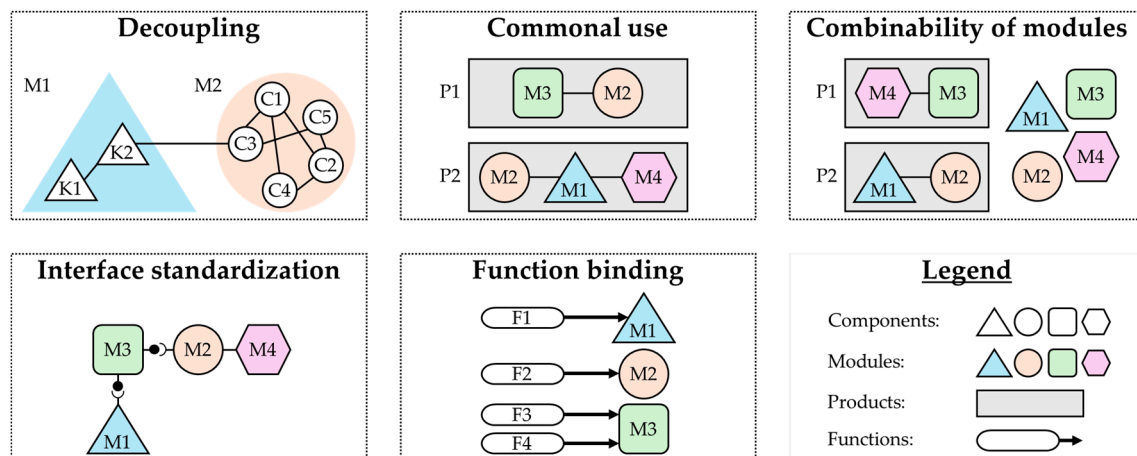


Figure 3. Characteristics of modularity adapted from [19,20,26].

Communal use facilitates economies of scale through reuse across multiple products, while combinability enables the realization of product variants through alternative module configurations. These properties are operationalized through standardized interfaces ensuring compatibility and functional binding assigning clearly defined responsibilities to modules.

In this context, modularity is best understood as a gradual property of product structures, determined by the extent to which interactions are internalized within modules. Modules thus represent coherent groupings of components with strong internal relationships and can be treated as coherent logical units [20].

The objective of MPD is not to maximize modularity per se, but to tailor modular characteristics in a way that yields lifecycle-spanning benefits aligned with company- and product-specific goals. A key advantage lies in the ability to configure product variants through module combinations, thereby achieving high external variety while reducing internal complexity. As these benefits unfold primarily at the level of multiple variants, modularity should be designed and evaluated from a product family perspective rather than for isolated products.

Product modularization has been addressed through a variety of methodological approaches [27], each emphasizing different objectives, lifecycle phases, and methodological concepts. Despite this diversity, most approaches follow a common procedural logic in which an existing product structure is captured, analyzed, and reorganized into modules according to defined criteria. This process typically comprises four core steps, as suggested by [20]:

1. Decomposition of the product structure, in which the product is broken down into its constituent components.
2. Analysis of components and relationships based on technical, functional, and strategic criteria.
3. Grouping of components into modules according to selected modularization drivers.
4. Synthesis of a modular product structure including the definition of module boundaries and interfaces.

While widely adopted, modularization approaches vary in their methodological realization and tool support. Moreover, the integration of the resulting modular structures into organizational processes and data landscapes is often insufficiently addressed in existing work. A critical enabler of modularization is the decomposition of the product into sufficiently granular elements, allowing for a detailed assessment of dependencies and trade-offs. During this process, conflicting objectives—such as cost efficiency, product differentiation, and lifecycle optimization—must be systematically balanced. The outcome is a modular product structure comprising well-defined modules and interfaces, which must be reintegrated into enterprise processes and data structures. Once established, modules can serve as standardized units for cross-functional coordination, thereby reducing both organizational and technical complexity.

While variability management is closely related to modularization, the literature largely remains at a conceptual level, with limited methodological guidance for the systematic generation of variants.

3. Related Work

As discussed in the previous section, the fields of variability management and SPLE, as well as MPD, are well-established research domains. Consequently, a considerable body of research exists within these areas, including studies that attempt to combine both perspectives. During our literature review, we observed that research combining product lines and MPD often naturally reflects the disciplinary background from which it

originates. Contributions rooted in mechanical or product engineering typically emphasize MPD, whereas work originating from software engineering and information systems tends to focus more strongly on SPLE. Depending on the research origin, one of these perspectives is usually emphasized over the other. For this reason, we first discuss research that connects both approaches but primarily originates from the mechanical engineering domain, followed by work stemming from the software engineering perspective.

Within the first group of studies, it is widely recognized that variability management is necessary to effectively govern product modules; without such mechanisms, modules would exist largely as isolated elements without systematic coordination. For instance, Blees et al. [28] present an approach that combines design for variety (which has the goal of reducing the impact of variants on the life-cycle costs of a product [29]) with MPD and apply a variety allocation model to a product family. However, product line engineering (PLE) is not explicitly addressed, and the work does not demonstrate how very large numbers of modules can be systematically managed. A different perspective is presented by Berschik and Krause [30], where the authors discuss the combination of PLE and product modularization within a model-based systems engineering (MBSE) framework. The approach proposes linking modules of product families to specific product variants, illustrated through the example of robotic vacuum cleaners. Nevertheless, the application of core PLE principles remains rather abstract, and the approach does not address the scale of variability encountered in the automotive industry, where the number of variants offered and produced can grow exponentially. A similar tendency can be observed in other studies originating from the mechanical engineering domain (e.g., [31–33]). These approaches typically rely on simplified examples with a limited number of variants and do not explicitly apply PLE concepts. As a result, they are difficult to transfer to the automotive industry, where product structures are significantly more complex and the number of variants is substantially higher.

At the same time, research originating from the (S)PLE communities has explored related topics from a different perspective. For example, several studies investigate multi-product lines [34–36], i.e., the combination of multiple product lines that each represent individual systems but must interact with other product lines within a larger system context. While these approaches address the coordination of multiple product lines, they generally do not explicitly discuss modules or their systematic control through PLE mechanisms.

However, applying such concepts specifically to the automotive industry requires a deeper understanding of the artifacts and structures through which modules are controlled. This, in turn, requires a clearly defined PLE structure at the level of the entire vehicle system. Research addressing this level of abstraction remains scarce. Although some studies [37,38] discuss product line concepts at the vehicle level, they typically do not establish a direct link to MPD.

Another related research area concerns the integration of MBSE and PLE, including approaches for configuring MBSE artifacts within product lines [39–42]. While this work contributes valuable insights into model-driven configuration and system engineering practices, it likewise does not address which concrete variability characteristics or features should be used to control modules in large-scale automotive product lines.

A notable observation is that only limited research exists on the combined application of MPD and PLE that specifically addresses the requirements of the automotive industry. On the one hand, there is a substantial body of literature discussing modularization in the automotive domain. Examples include studies on modular platforms, modular product architectures, and manufacturing networks in the automotive industry (e.g., [43–47]). These works analyze the role of modularization in product development and production but generally do not explicitly incorporate PLE principles. On the other hand, several studies

investigate the application of SPLE in the automotive context (e.g., [37,38,48]). In particular, some research [49,50] characterizes the automotive domain as a form of “mega-scale PLE” due to the enormous number of variants and the scale of production.

However, the integration of SPLE and MPD in automotive systems remains largely unexplored. Existing research lacks approaches that address domain-specific requirements and enable systematic control of modules and system elements in large-scale product lines, despite the growing complexity of automotive variability.

4. Research Approach

To investigate how SPLE can be extended and systematically integrated with MPD for effective module management in automotive systems, this research follows the design research methodology (DRM) as proposed by Blessing and Chakrabarti [51]. The overall methodology and its alignment with the research questions are structured according to the DRM framework and illustrated in Figure 4.

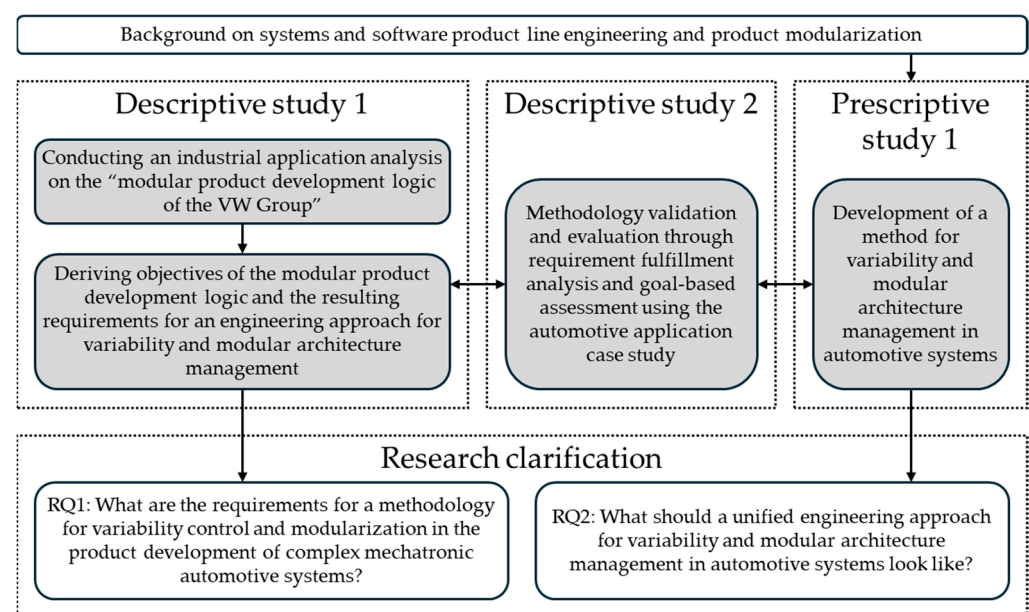


Figure 4. Methodology and approach for answering the research questions.

The research process begins with the research clarification phase, in which the central research questions (RQs) are defined. The study addresses the following two RQs:

- RQ1: What are the requirements for a methodology for variability control and modularization in the product development of complex mechatronic automotive systems?
- RQ2: What should a unified engineering approach for variability and modular architecture management in automotive systems look like?

To answer these RQs, the research is structured into sequential studies following the DRM logic. The first phase, descriptive study 1, establishes the theoretical and empirical foundation. It includes a structured analysis of SPLE as well as product modularization. In addition, a detailed industrial analysis within a proof of concept is conducted, focusing on the MPD logic applied within the Volkswagen Group. Based on this analysis, the underlying objectives of this MPD logic are systematically derived. These objectives are subsequently translated into a set of requirements for an engineering approach that integrates variability management and modular architecture design. This phase primarily contributes to answering RQ1.

Building on these requirements, the prescriptive study 1 focuses on the development of a unified methodology for variability and modular architecture management in automotive systems. The proposed methodology integrates principles from SPLE with concepts from MPD, leveraging the theoretical foundations established in the previous phase. The resulting methodology aims to enable consistent management of variability and modular structures across system levels.

The developed methodology is subsequently evaluated within descriptive study 2 through a combined validation and evaluation approach based on requirement fulfillment analysis and goal-based assessment using the automotive proof of concept. The assessment examines the extent to which the proposed methodology satisfies the requirements previously derived in descriptive study 1 while simultaneously evaluating its capability to support the identified objectives of the MPD logic within the investigated automotive context. This evaluation provides an initial indication of the applicability and effectiveness of the proposed approach for variability and modular architecture management in complex automotive systems.

Figure 4 provides an overview of the entire research process, highlighting both the DRM-based structuring of the study and the relationships between individual research activities and the RQs. It further illustrates the interdependencies between the different phases and their respective contributions to the overall research objectives.

5. Automotive Context and Methodological Requirements

This chapter is structured as follows. Section 5.1 presents the automotive application context using the example of the Volkswagen Group, highlighting the need for systematic module control in the context of MPD. Section 5.2 builds upon this analysis by deriving goals from the described application, an industrial analysis conducted within the Volkswagen Group in the context of the ongoing transition towards modular product development and the group systems store (GSS), complemented by exemplary strategic goals of Porsche AG. The analysis was based on internal project activities, existing product documentation processes, strategic objectives, and a series of discussions with experts from the fields of systems engineering, product documentation, modular architectures, and variability management. Based on these inputs, market- and technology-related objectives were identified and consolidated. These objectives subsequently served as the basis for the derivation of the methodological requirements presented in Section 5.3.

Based on these objectives, concrete requirements for a supporting methodology are defined in Section 5.3. The analysis further demonstrates that existing approaches, namely SPLE and MPD, when considered in isolation, are not sufficient to fulfill these requirements in a comprehensive manner.

5.1. Automotive Case

The automotive case is exemplarily derived from the brand strategy of Porsche AG, a subsidiary of the Volkswagen Group. However, the objectives defined for this case are aligned across all brands within the Volkswagen Group and are therefore equally applicable at the group level. Within the framework of Porsche's Strategy 2030 Plus [52], the pillars customer, product, and transformation define clear expectations for how future technologies must be developed, deployed, and integrated across product lines. From a market perspective, Porsche responds to intensifying global competition, accelerating technology cycles, and rising customer expectations by emphasizing technological leadership, rapid innovation adoption, and the creation of flexible, value-oriented product offerings. At the same time, the product strategy calls for a scalable and future-proof portfolio, enabled by

car software, key technologies, and efficient development processes that support faster cycles and reduced complexity.

As a solution approach, the GSS directly addresses Porsche's demand for adaptive product architectures by operationalizing the principles of MPD at the system level. Its primary objective is to establish reusable systems and modules as stable, configurable building blocks that can be systematically applied across product lines, platforms, and brands. In doing so, the GSS translates the theoretical foundations of MPD into a system-centric and operational development framework.

Within this framework, the GSS enables the structured definition, development and reuse of modular mechatronic systems independently of individual vehicle projects. Modules are treated as members of system families with explicitly defined scopes and controlled variability, allowing technological advancements to be propagated consistently across multiple product derivatives.

The current product documentation within the Volkswagen Group is structured around hardware platforms and E/E platforms. In automotive engineering, hardware platforms have long served as a foundation for managing product complexity by facilitating the standardized reuse of components, resulting in reduced development effort, lead time, and engineering costs [53]. A hierarchical three-level product classification system is employed, comprising the platform class, the vehicle group class, and the vehicle class (or "Hat"). At the platform level, vehicles from different brands are consolidated according to predefined criteria, following common-part strategies that vary by vehicle type. Within each platform, the vehicle group class clusters vehicles that meet additional criteria, such as identical design responsibility within a brand or a high degree of commonality across vehicle classes. Different derivatives of a vehicle are assigned to a single vehicle class when the effort required to manage variability is lower than that of maintaining an additional stand-alone class.

To simplify the maintenance of assembly structures independently of specific products within a vehicle class, the organization currently relies on development and production toolkits. These toolkits aim to prevent the redundant maintenance of sub-structures across the corresponding structural bill of materials (BOM).

Figure 5 illustrates the concept of a hardware platform as a shared foundation that is complemented by vehicle-specific design elements, commonly referred to as the "hat" [53]. An E/E platform represents a common collection of reusable E/E components from which vehicle-specific E/E architectures can be systematically derived [54]. The E/E platform incorporates software artifacts associated with both the hardware platform and the hat, together with their physical realization in electronic control units (ECUs), thereby forming an interconnected embedded system [53]. The underlying modular toolkit and platform-hat logic is currently predominantly aligned with vehicle- or vehicle-family-specific product development processes. In this paradigm, platforms primarily serve as enablers for individual vehicle projects, while systems and modules are instantiated, adapted, and validated within the context of a specific hat.

With the increasing complexity of software-intensive and mechatronic vehicle systems, this project-centric orientation reveals inherent limitations, particularly with regard to system reuse, controlled variability, and lifecycle-spanning governance. Consequently, there is a growing need for system- and platform-centric development approaches that decouple system definition from individual vehicle projects.

Looking ahead, the Volkswagen Group is strategically transitioning toward a MPD logic. This shift forms the basis for solution concepts such as GSS. At the highest level, this logic comprises three phases: the strategy phase, the technology provisioning phase, and the Hat-PEP (product emergence process) phase, as shown in Figure 6. In the strategy

phase, stakeholder requirements from customers, society, business units, and model lines are consolidated independently of any specific product or project. This consolidated requirements baseline guides subsequent technology development.

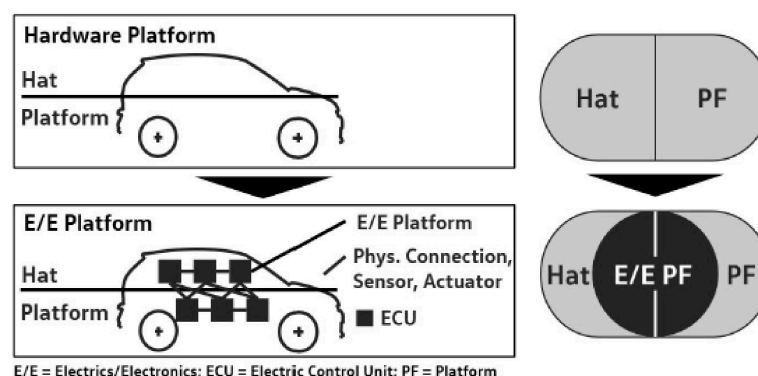


Figure 5. The platform concept [53].

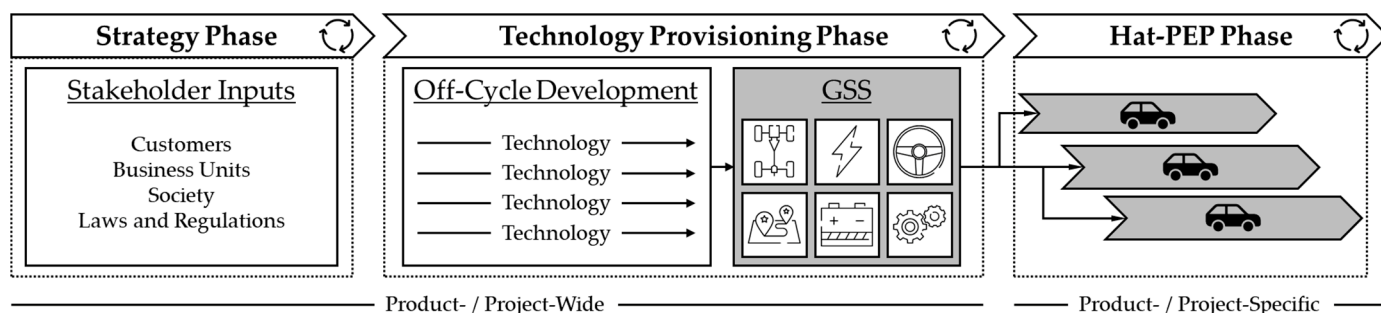


Figure 6. Modular product development logic including the GSS.

The technology provisioning phase covers the development of mechatronic systems and their assemblies. This phase is characterized by decoupled technology development, commonly referred to as off-cycle development. In contrast to traditional project-bound vehicle development, off-cycle technology provisioning aims to develop, validate, and deliver technologies independently of specific vehicle projects and across brands. During the Hat-PEP phase, vehicles draw on the modular systems developed during the technology provisioning phase, applying them in a product- and project-specific manner.

This aligns with the broader industry shift toward modular, reusable system architectures and cross-platform technology sharing, as also emphasized in previous research [55]. Technology provisioning is operationalized through GSS, which provides a digital representation of the solution space spanning platforms, derivatives, and technologies. The GSS is envisioned to establish the central structuring logic for reusable systems and modules across all product lines. This approach is expected to enable vehicle projects to draw on pre-developed and validated systems and system composites, thereby reducing time-to-market, creating synergies, and lowering complexity. To support this objective, the organization has defined a sequence of steps for system development within the technology provisioning phase, addressing multiple use cases, as illustrated in Figure 7.

“Define the System” starts with the specification of system-level requirements. This includes the definition and management of logical system components as well as the assignment of system functions. A logical system component is characterized by technical attributes derived from the overarching functional architecture. As the process progresses, logical variants of the system component are defined and described using a central feature catalog. This catalog provides a harmonized vocabulary covering both technical and commercial properties, ensuring consistent and reusable descriptions across domains.

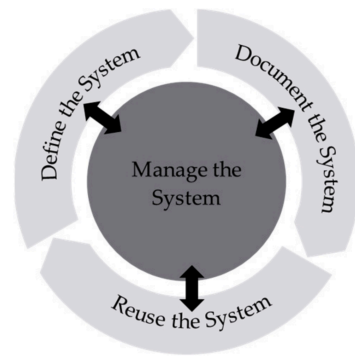


Figure 7. Sequence of steps for the technology provisioning phase.

“Document the System” focuses on the realization of logical system components through concrete technical solutions. In this step, logical elements are mapped to their corresponding physical, electronic, or software realizations, resulting in the creation of implementation-specific engineering artifacts. Physical system components are described using technical attributes maintained in the central feature catalog, ensuring consistency between logical definitions and their technical realization. A key aspect of this step is the specification of temporal validity and release states, which establishes the basis for subsequent configuration, variability, and lifecycle management activities.

“Reuse the System” addresses the deployment and reuse of systems within vehicle projects, including Hat classes and platforms. In this step, vehicle projects [53] integrate the approved systems or system composites while considering the temporal validity and technical feature specifications relevant to the respective project context. This ensures that reused systems are consistently embedded in project-specific architectures while maintaining alignment with defined release states and configuration constraints.

“Manage the System” encompasses the controlled modification of systems and system composites through traceable change-management processes, complemented by maturity and release management. The primary focus of this step is the management and governance of variability, both at the logical system-component level and at the level of physical realization. In modern vehicle development, variability and release management have become essential capabilities, particularly in development contexts characterized by increasing cross-domain variability.

In this context, the GSS aims to provide a unified, cross-domain structuring and management logic that supports the reuse, evolution, and controlled variability of mechatronic system modules. This capability aligns directly with the market- and technology-driven priorities defined in Porsche’s Strategy 2030 Plus [52], which emphasize technological leadership, efficient development and flexible product architectures.

5.2. Definition of Goals

The objectives pursued in the research field of MPD (see Section 2.2) are largely aligned with those of the GSS. Building on the organization’s overall corporate objectives, specific goals for the GSS are derived in the following. These goals can be classified into market-related and technology-related objectives. In the remainder of this section, these objectives are systematically assigned, indexed, and briefly described.

Table 1 provides an overview of the market-related and technology-related objectives derived from industrial project work and discussions with experts in the field of product documentation. They form the derived goal set for a system governance approach and are grouped into two categories: market (M-01–M-06) and technology (T-01–T-07).

Table 1. Goals for modular mechatronics systems design.

Category	ID	Objective	Description
Market	M-01	Market expectations	Enable the continuous adaptation of systems to changing customer expectations to sustain competitiveness and value
	M-02	Technology trends	Enable the integration of emerging technologies into system architectures to ensure relevance and innovation speed
	M-03	Technological leadership	Enable technological leadership in key domains affecting performance and digital experience
	M-04	Product portfolio planning	Enable scalable and reusable modules to support flexible portfolio development across product lines
	M-05	Demand-driven development	Derive system scope from validated customer needs and actual usage patterns
	M-06	Service orientation	Enable lifecycle services, including connectivity and digital services
Technology	T-01	Decoupled, off-cycle system development	Enable off-cycle technology maturation to support reuse and roadmap alignment
	T-02	Explicit variant creation & governance	Establish clear and systematic mechanisms to create, describe, and control system variants across domains
	T-03	Networked mechatronic development	Integrate hardware, software and E/E domains with software as the central system integrator
	T-04	Technology focus & competence centers	Consolidate critical expertise in competence centers to ensure architectural consistency
	T-05	High product maturity & module release	Modules must reach maturity early and be releasable without direct usage context
	T-06	Continuous system development & integration	Enable continuous integration and testing to ensure system coherence throughout development
	T-07	Vehicle Hat provisioning/integration	Minimize integration effort for vehicle projects through reusable and pre-validated modules

5.3. Requirements Definition

Derived from the objectives identified in the automotive application context, the following section defines the specific requirements for an integrated methodology. The requirements are derived by systematically refining the goals identified in Section 5.2 into methodological capabilities required to support MPD and variability management. Following the approach proposed by Visser et al. [56], and the requirement formulation principles described by Bender and Gericke [57], the goals are translated into functional requirements and reviewed against the industrial use case. This ensures traceability between industrial objectives, methodological requirements, and the subsequent evaluation of the proposed methodology.

The focus is placed primarily on functional requirements. While a comprehensive methodology would naturally also need to address non-functional requirements [58], these cannot be reliably assessed within the scope of an initial proof of concept and are therefore not considered in detail [56].

The resulting requirements for the methodology are summarized in Table 2. These requirements are explicitly linked to the objectives outlined in the previous section, thereby ensuring traceability to their origin. The goals, adapted to the specific use case, have been further refined and operationalized into the resulting requirements.

Table 2. Requirements for an integrated methodology.

ID	Requirement	Source	SPLE	MPD
R-01	The methodology shall represent mechatronic, hardware, or software systems.	M-03, M-06, T-03, T-06	●	●
R-02	The methodology shall support the representation of RFLP ¹ layers.	M-02, T-03, T-06	●	●
R-03	The modules shall be uniquely and unambiguously describable.	M-04, T-02, T-04	●	○
R-04	The modules shall rely on a consistent description methodology.	M-04, T-02, T-04	○	●
R-05	The modules shall be combinable within a system architecture.	M-04, T-01, T-03, T-06	●	●
R-06	The modules shall support the representation of relationships to other modules.	M-04, T-02, T-06	●	●
R-07	The modules shall be describable through defined properties and characteristics.	M-05, T-02	●	○
R-08	The modules shall provide clearly defined interfaces.	M-04, M-06, T-01, T-03, T-07	○	●
R-09	The modules shall be configurable based on system requirements.	M-01, M-05, T-02, T-06	●	○
R-10	The modules shall be representable in multiple variants.	M-01, M-04, T-02, T-03	●	○

●: Requirement satisfied; ●: Requirement partially satisfied; ○: Requirement not satisfied. ¹ RFLP: Requirement, Functional, Logical, and Physical.

Table 2 also illustrates the comparison with the research streams introduced in Section 2, which the proposed methodology aims to integrate. The analysis clearly shows that neither of the existing approaches is able to satisfy all identified requirements in a unified manner. However, achieving precisely this integration is essential and constitutes the primary objective of this paper. This discrepancy becomes particularly evident when considering two specific requirements. Requirement R-07 addresses how modules should be described and controlled, an aspect for which MPD does not provide a clear answer. Conversely, requirement R-04 focuses on the definition of modules, a topic that is not explicitly addressed within SPLE.

However, achieving precisely this integration of research streams and methodologies is essential for the presented automotive use case and constitutes the primary objective of this paper.

6. Methodology

In this section, the proposed modular systems product line engineering (MSPLE) methodology is introduced, which aims to enable holistic MPD within the automotive domain. The methodology combines principles from SPLE and MPD to address increasing system complexity and variability across automotive system domains. The underlying analysis has shown that neither SPLE nor MPD alone is sufficient to fulfill the requirements of the automotive application context. Instead, their systematic integration enables a comprehensive approach for managing variability and modular architectures. Furthermore, the MSPLE methodology builds upon the RFLP (requirement, functional, logical, and physi-

cal) framework [59] and incorporates systems engineering principles to ensure consistent structuring, traceability, and synchronization of engineering artifacts and processes.

The four phases of the automotive application context, previously shown in Figure 7, are explicitly represented within the MSPLE methodology, illustrating how variability is structured, controlled, and resolved across the RFLP layers. In doing so, the methodology emphasizes the systematic steering of systems through configuration-relevant attributes and decisions, which forms the core focus of this research.

An overview of the MSPLE methodology is provided in Figure 8.

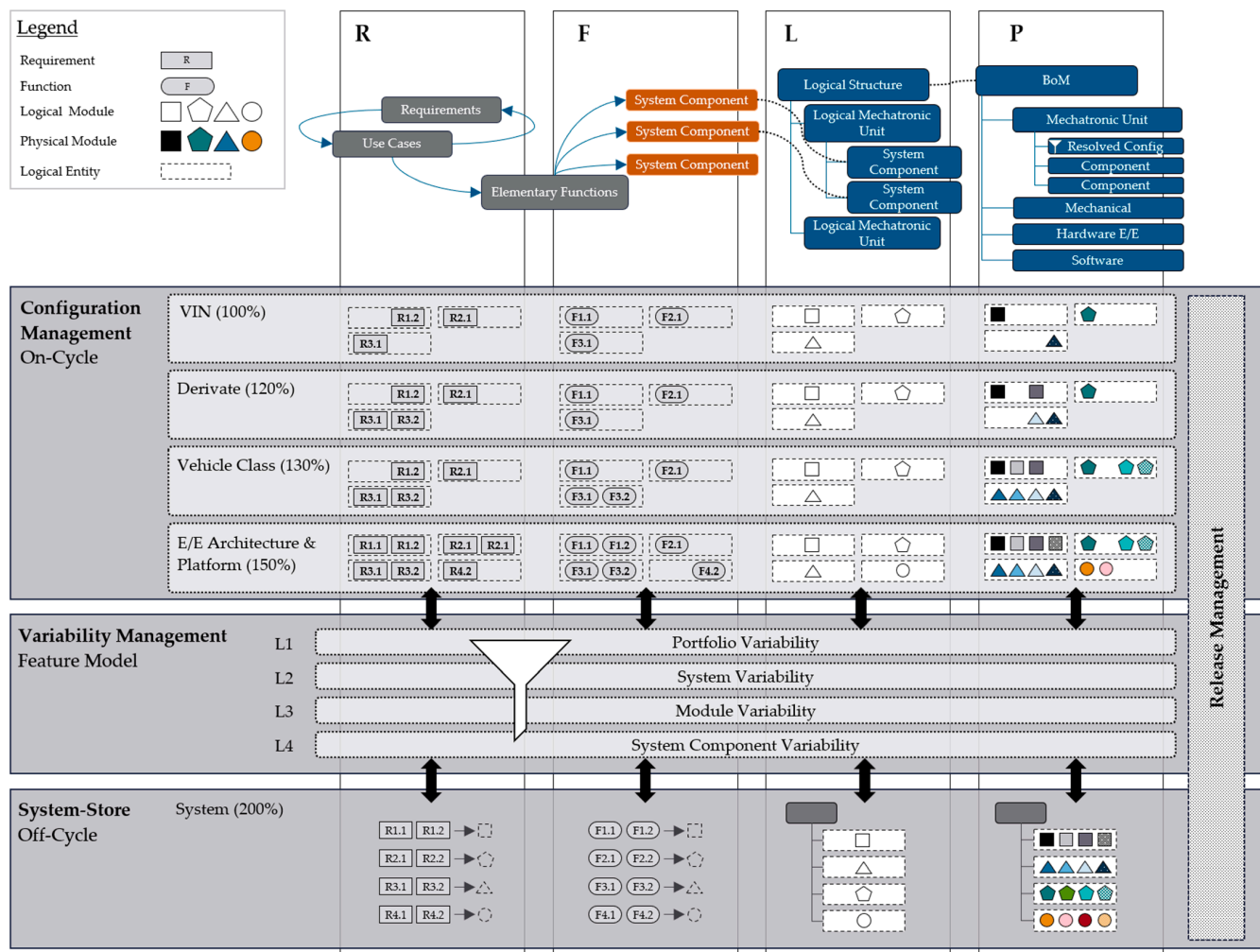


Figure 8. MSPLE methodology for modular management of mechatronic systems in automotive product lines.

The methodology is structured into several distinct areas. First, the RFLP layers represent the artifacts of the solution space, capturing requirements, functional, logical, and physical elements. Second, a dedicated feature modeling layer describes the variability of the system. In addition, two operational domains are distinguished: the off-cycle domain and the on-cycle domain. The off-cycle domain comprises elements such as the system store or module catalog, in which reusable artifacts are developed, managed, and maintained independently of specific product instances. In contrast, the on-cycle domain focuses on the application of these artifacts within concrete contexts, such as platform definitions or individual vehicle projects, where configuration decisions are made and artifacts are instantiated accordingly.

Subsequently, the structures and workflows of the MSPLE methodology are described in greater detail, and the elements of the model are explained in relation to the automotive application context. The defined phases of the GSS are mapped to the corresponding layers of the RFLP model, thereby illustrating how features, variants, and configuration options are consistently linked across requirements, functional, logical, and physical representations. Particular emphasis is placed on managing the system, where a conceptual solution for variability-driven system steering and configuration management is presented and discussed.

Before discussing the individual MSPLE phases in detail, an overview of the overall workflow is provided in Table 3, including the inputs, outputs, activities and responsible roles of each phase. This overview is intended to facilitate understanding of the methodology and provide a structured context for the detailed phase descriptions that follow.

Table 3. Overview of the phases in the MSPLE methodology.

Phase	Input	Key Activities	Roles	Outputs
Define the System	System Scope	Requirements, Functional & Logical Realization	Requirements Engineer, Architect	Requirements, Functional & Logical System Architecture
Document the System	Logical Architecture	Physical Realization	Domain Engineers	Physical Artifacts, Documented Modules, Variant Descriptions
Manage the System	System Artifacts, Modules	Feature Modeling, Dependency Definition, Variability Governance	Product Line Architect	Feature Models, Variability Rules
Reuse the System	Modules, Feature Models, System Artifacts	Product Configuration, Variant Derivation	Product/Project Teams	Product-Specific System Configurations

6.1. Phase: Define the System

The objective of the “Define the System” phase is to establish a requirements- and function-oriented system description that enables the modeling of functional-logical effect chains within the RFLP framework. Requirements serve as the representation of the problem space, from which use cases are derived and decomposed into elementary functions. These functions are allocated to system components in the logical layer, with one or more system components constituting a mechatronic system as a modular architectural unit.

Figure 9 presents a detailed excerpt of the overall model and focuses on the “Define the System” phase. At the lowest level, the system store layer is represented, which corresponds to a 200% system level realized in an off-cycle context. In this context, off-cycle means that systems are defined and, in subsequent steps, documented independently of any concrete usage or vehicle project.

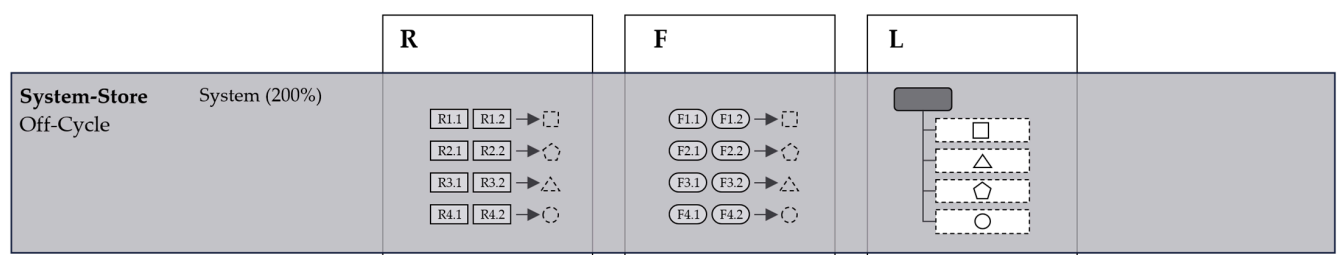


Figure 9. Phase: define the system.

At the requirements layer, stakeholder needs are first systematically captured. These requirements may be either functional or non-functional in nature. The transformation process involves translating informal, natural-language descriptions into a formalized language with defined notation and parameterization. In accordance with a systems-of-systems perspective, the requirements specified at the R layer are decomposed across the different system levels (L1 to L4).

Across Levels 1 to 4, requirements are specified for the overall system and subsequently allocated to mechatronic modules and systems. The functions required to satisfy these requirements are realized within mechatronic systems and mapped to the corresponding logical entities, thereby establishing the foundation for modular system structures.

Figure 9 exemplarily illustrates requirements (R) that influence mechatronic modules and systems within the system store. The symbols representing modules and systems are depicted with dashed outlines, as the logical delineation of the system is performed only in a subsequent step within the logical architecture layer (L).

At the functional layer (F), use cases are derived from user stories and system requirements (R) and are used to describe the intended system behavior. These use cases are further decomposed into elementary functions. Elementary functions describe, in a solution-neutral manner, what the system must accomplish, independent of any concrete technical realization. Within the functional architecture, the elementary functions are decomposed into functional elements. This functional architecture forms the basis for the subsequent allocation of functions to logical system components. Analogous to the requirements layer (R), a dashed module symbol is also used in the functional layer (F) to represent the relationship between functions (F1.1–F4.2) and modules or systems.

In the logical architecture layer (L), the solution-neutral elementary functions defined in the functional architecture are systematically allocated to the logical elements responsible for their realization. This traceability ensures that each function is consistently assigned to one or more logical system components without prematurely introducing implementation decisions. Within the logical architecture, one or more system components together constitute a module or system as defined by the GSS. Consequently, the symbol representing a logical module or system is depicted with a solid outline at this layer.

In addition to defining and representing logical elements, the logical layer (L) also provides a system structure. Within the Volkswagen Group, this structure is established by the logical reference system architecture (LRSA), which defines a project-independent, binding, and cross-brand logical system hierarchy. Derived from the functional architecture, a system component currently represents the smallest cross-brand aligned logical architectural element (L4) and serves as a key building block for modular system structuring. By bundling related functions and responsibilities, system components support a modular decomposition of the overall system.

Looking ahead, logical mechatronic units are envisaged as an additional, controllable logical layer for modularization, allowing a more flexible, cross-domain, and reusable organization of module and system functionality. The logical system definition thus provides the foundation for the subsequent transition to the physical realization layer.

6.2. Phase: Document the System

The physical realization layer (P) aims to map the logical elements from the logical layer (L) to concrete mechanical, electronic, or software-based implementations. These implementations are represented by dedicated engineering artifacts such as drawings, source code, or 3D models. As part of the modularization strategy, the P layer is extended to document mechatronic units in addition to domain-specific engineering artifacts.

As outlined previously, development and documentation are conducted independently of concrete vehicle projects. Modules and systems are integrated into the overall structure as resolved configurations, representing fully specified solution-space instances derived from variability decisions. These off-cycle developed modules and systems are subsequently complemented by vehicle-specific integration scopes, which capture application-engineering adaptations required for a particular project context. Figure 8 illustrates these integration scopes as components within the resolved configuration, thereby explicitly distinguishing reusable off-cycle system assets from vehicle-specific application adaptations while preserving traceability across the RFLP layers.

The system store builds upon the LRSA structure, which serves as the organizational framework for realized system components and mechatronic units. Figure 10 provides a detailed view of the “Document the System” phase. In this phase, the structure and form are inherited from the logical layer, ensuring continuity between the logical system definition and its subsequent documentation. The logical architecture thus serves as the authoritative reference for organizing and structuring the documented system elements, while the focus shifts from solution-neutral definitions toward the systematic documentation of realizable system content within the solution space.



Figure 10. Phase: document the system.

Within the logical solution space of the respective module, variants with specific characteristics are now represented. These variants reflect alternative realizations within the module or system solution space and capture differences in functional, technical, or configuration-relevant properties. Variants within the module or system solution space are visually distinguished by color coding.

At the physical realization level, modules and systems are described using technical attributes maintained in the central feature catalog, providing the basis for systematic variability and configuration management. This feature catalog establishes a binding semantic space for describing variability and encompasses both technical characteristics and commercial attributes (e.g., markets or regions). To ensure effective variability and configuration management during the “Reuse the System” phase, all system levels must be controllable through technical attributes representing variability drivers and constraints.

The proposed approach builds on attribute-based controllability, where technical attributes explicitly represent variants and variability drivers, enabling consistent definition, propagation, and evaluation of variability throughout the entire development lifecycle. Applied to the representation of this phase, this implies that all colored elements are systematically attributed with parameters derived from the central feature model. These parameters explicitly capture variability drivers, constraints, and configuration-relevant properties, thereby linking the models/systems to the underlying variability model. As a result, the represented elements are no longer static architectural entities but become configurable assets whose instantiation can be controlled through configuration decisions. This attribution enables the elements to be consistently reused in subsequent phases in

accordance with concrete configuration requests, while preserving traceability between feature selections, variant resolution, and the corresponding system representations.

6.3. Phase: Manage the System

The objective of this phase, shown in Figure 11, is to establish the foundation for the systematic reuse of existing artifacts, including requirements, functions, and previously defined modules. Achieving this requires a clear linkage to features, which serve as the decision mechanism for determining under which conditions these artifacts can be reused.

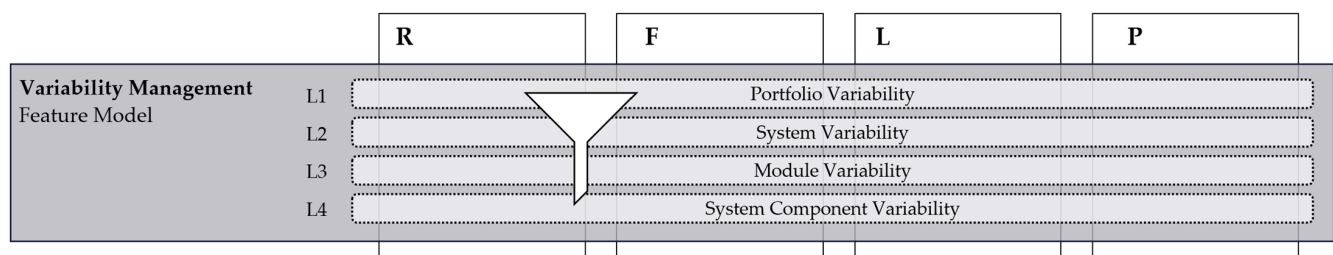


Figure 11. Phase: manage the system.

Accordingly, this section addresses two key aspects. First, it outlines the general structure of the feature models and the types of information they must capture. Second, it describes how artifacts need to be linked to these features in order to enable their consistent and scalable reuse. To this end, the MSPLE methodology leverages established principles from SPLE. From a structural perspective, feature models must represent the variability of both the artifacts and the vehicle itself, thereby reflecting diverse customer requirements. This requires modeling the vehicle topology in a structured manner. Following systems engineering principles [60], the overall system is decomposed into hierarchical levels, resulting in multiple feature models across different system layers. Each model captures the variability of a specific system level, ranging from subsystems to individual components, using dedicated feature sets.

A key design decision is the avoidance of a monolithic feature model. Instead, each subsystem maintains its own feature model, with clearly defined responsibilities and boundaries. At lower system levels (e.g., L2 to L4), the identification of variability is comparatively straightforward, as it is largely determined by the inherent characteristics of the respective subsystems. While defining appropriate features still requires careful consideration, the scope remains manageable.

In contrast, variability modeling at the highest system level (L1) is significantly more complex. Here, the entire vehicle must be considered from multiple stakeholder perspectives. Technical development defines system-level variability from an engineering standpoint, while product lines determine which features are included in specific vehicle configurations. Marketing introduces dependencies between features and defines offering structures, whereas customers ultimately configure vehicles according to their individual preferences. All of these perspectives must be consistently represented within the L1 feature model.

To address this complexity, the MSPLE methodology distinguishes between two types of features at the L1 level. The first type represents capabilities, i.e., the functional characteristics of the vehicle that define what the system is able to perform. These capabilities form the primary basis for configuration decisions by stakeholders, including end customers. The second type captures structural data, which represents contextual or boundary conditions that also introduce variability but are not always directly or explicitly selectable. Examples include the market of delivery, production location, or vehicle brand. These structural features influence the availability and behavior of other features and are therefore

essential for a consistent configuration logic. Within the L1 feature model, both types of features must be interconnected through relationships and constraints. These dependencies ensure that valid configurations can be derived while allowing different stakeholders to make consistent and meaningful selections.

Once feature models are established across all system levels, an additional prerequisite for reuse is the definition of relationships between these levels. To ensure that these relationships remain manageable, particularly with respect to the temporal evolution of the system, a consistent rule for defining dependencies is required.

The MSPLE methodology enforces a bottom-up approach, in which relationships and constraints are defined exclusively from lower system levels toward higher levels. This design decision is motivated by two main considerations. First, responsibility for variability knowledge resides primarily at the level where artifacts are developed; thus, system owners are best positioned to define the dependencies of their modules. Second, this approach prevents inconsistencies at higher system levels, where stakeholders may otherwise define conflicting constraints without full visibility of underlying dependencies.

In summary, the MSPLE methodology establishes a structured framework for feature modeling across multiple system levels. By systematically providing features as the basis for representing variability within artifacts, particularly within modules, it enables the consistent and scalable reuse of development assets in a complex and evolving system landscape.

Once these significant upfront investments have been made and a comprehensive system of feature models has been established, the focus of this phase shifts toward their continuous maintenance and evolution. This includes keeping the models consistent, up to date, and aligned with ongoing development activities.

Another key objective of this phase is the systematic linkage of features to artifacts in the solution space, as illustrated in Figure 8, with a particular focus on modules. In this context, artifacts typically exhibit a n -to- n relationship with features, as the selection of a specific artifact, e.g., a module, may depend on the combination of multiple features.

A key requirement in this context is that every evolutionary change in the underlying artifacts must be consistently reflected in the corresponding feature models. Only in this way can the models accurately represent the current state of the solution space and avoid the emergence of obsolete or dead features. However, the integration of temporal control mechanisms across the entire product lifecycle is not addressed within the scope of this research and therefore represents a limitation of the presented MSPLE methodology.

6.4. Phase: Reuse the System

The phase “Reuse the System”, shown in Figure 12, describes the systematic reuse of system artefacts that have been defined and structured in the preceding phases of the MSPLE methodology. Artefacts—including requirements, functions, and previously defined modules—are linked to features and thereby organized in a hierarchical structure and interconnected across the system levels L1 to L4. This structured representation establishes a consistent, cross-domain model that serves as the foundation for all subsequent reuse and configuration activities.

Within the context of reuse and configuration processes, multiple stakeholders must be considered, including technical development, requirements engineering, test management, and the end customer. These stakeholders pursue different objectives with respect to the configurations they create and therefore impose heterogeneous requirements on the configuration process. For instance, engineering stakeholders may focus on technical feasibility and consistency, while management stakeholders are primarily concerned with portfolio structuring and reuse efficiency.

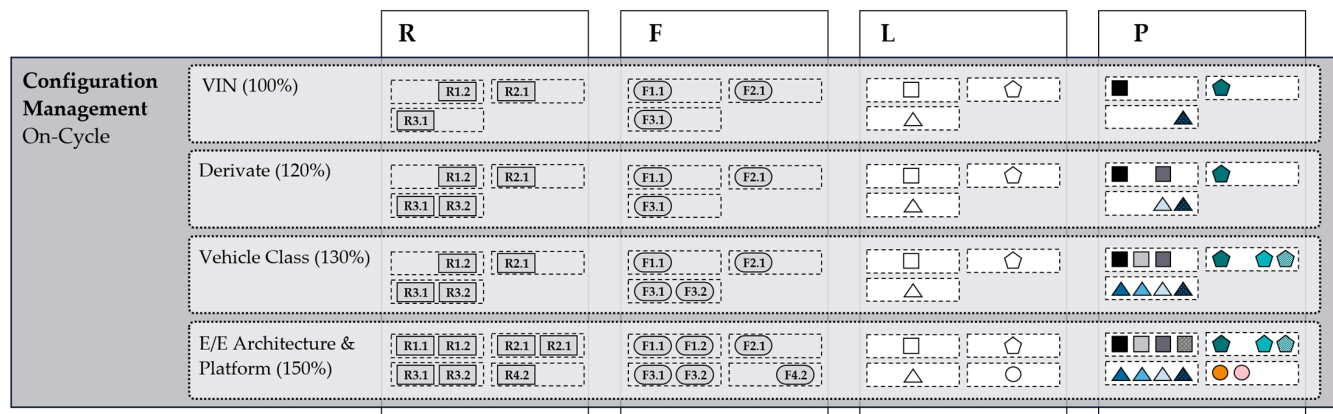


Figure 12. Phase: reuse the system.

To address these differing requirements, the proposed methodology structures the configuration process into multiple hierarchical levels, as shown in Figure 12. Starting from the system store (200%), configuration activities proceed through successive abstraction levels: E/E architectures and platforms (150%), vehicle projects (130%), vehicle derivatives (120%), and finally vehicle identification number (VIN)-specific or customer-specific vehicles (100%). At each of these levels, partial configurations are either defined or refined. This layered approach enables stakeholders to filter and access only those artefacts that are relevant to their specific context and decision-making scope.

Beyond stakeholder-specific filtering, the introduction of explicit configuration levels provides a significant conceptual advancement: it externalizes configuration knowledge that is typically only implicitly embedded within organizational processes. At each level, it becomes possible to explicitly reason about variability—i.e., which elements are variant and for what underlying rationale. Such explicit representation of variability knowledge, and particularly its centralized management, is largely absent in current industrial practice, where it is often distributed across tools and organizational units without formal integration [18].

Across all configuration levels—from the 200% system store down to the 100% VIN-specific or customer-specific vehicle—the RFLP logic is consistently maintained. This end-to-end traceability enables, for example, the unambiguous identification of requirements associated with a specific VIN or specific vehicle, thereby supporting targeted validation and test execution. Similar traceability benefits apply to the functional, logical, and physical domains, enabling a holistic and consistent engineering process.

The specification of vehicles from the 200% system store is achieved through the systematic creation of partial configurations. These configurations iteratively reduce the available degrees of freedom by progressively deselecting features. All configuration decisions are strictly governed by the rule base encoded in the feature models, ensuring consistency and validity throughout the derivation process. This mechanism enables a structured organization of the product portfolio and facilitates systematic reuse in accordance with SPLE principles.

In alignment with platform-based development paradigms, both E/E architectures and mechanical platforms (150%) are derived from the feature models defined in the system store (200%). The E/E architecture comprises multiple ECUs with predefined hardware and software configurations, thereby defining the execution environment for software-based functionalities. In contrast, the platform represents the mechanical structure and forms the physical basis for vehicle development. From the system store, multiple E/E architectures and platforms can be derived, reflecting alternative design decisions and variability constraints.

Each vehicle project (130%) is based on exactly one E/E architecture and one platform (150%). During project definition, non-required features are systematically deselected from both domains. The resulting reduced partial configurations are then merged into a consolidated configuration representing the vehicle project. This configuration captures the technical and functional scope at an intermediate level of specificity.

A vehicle project (130%) may serve as the basis for multiple vehicle derivatives (120%). For example, a single base model can be refined into several derivatives with distinct characteristics. The development of a derivative follows the same principle of configuration: features are further deselected from the vehicle project configuration, thereby increasing specificity and reducing variability. Each derivative thus represents a more concrete realization within the product line.

The transition from a vehicle derivative (120%) to a VIN-specific or customer-specific vehicle (100%) introduces a final configuration step in which all remaining variability is resolved. This step is typically performed by an end customer via a configuration interface, where features are selected from the set permitted by the derivative configuration. Importantly, the available feature set is strictly constrained by the previously defined partial configuration, ensuring consistency and feasibility. The resulting configuration is complete and unambiguous, with no remaining degrees of freedom, allowing the vehicle to be uniquely instantiated and produced.

It is important to note that the role of the “customer” in this context is not limited to the end user. Other stakeholders, such as test management, may also assume this role by configuring specific 100% vehicles for validation purposes. This enables the targeted generation of test instances that are fully traceable to underlying requirements and system artefacts.

Overall, in the “Reuse the System” phase, the defined feature space is systematically transformed into fully specified vehicle instances at the level of individual VINs. This transformation is governed by feature model constraints, structured across hierarchical configuration levels, and consistently linked via the RFLP paradigm, thereby enabling controlled variability management and end-to-end traceability.

7. Instantiation

In this section, the proposed MSPLE methodology is exemplarily applied using a case study from Volkswagen, depicting a proof of concept. The objective is to demonstrate how model-based development can be systematically implemented along the RFLP logic, how modular systems can be structured and composed, and how these systems can be linked to feature models to derive specific variants of artifacts and modules within the context of SPLE.

For the implementation and visualization of the example, mainly the 3DEXPERIENCE (v. R2026x) platform by Dassault Systèmes is used, enabling a MBSE approach. Requirements are modeled in Codebeamer (v. 2.2.1.4.3) by PTC and functional models are created using Cameo Systems Modeler (v. 2026x). The selected use case is a comparatively simple subsystem, namely the side (exterior) mirror.

To comprehensively illustrate the MSPLE methodology, the case study incorporates all relevant RFLP elements: requirements (R), functional models (F), logical structures (L), which already integrate modular concepts, as well as the corresponding physical and computer-aided design (CAD) models (P). These artifacts are consistently linked to a central feature model, which enables configuration and filtering based on specific features and system variants.

A particular emphasis is placed on the combination of multi-level modularization and SPLE principles. As a result, a configurable side mirror is realized, consisting of interchangeable modules that may themselves exist in multiple variants. This demonstrates

how variability can be consistently defined and propagated across different abstraction layers throughout the development process.

To exemplify the MSPLE methodology, the side mirror is initially modeled and subsequently structured as a modular system in the sense of a system of systems (SoS). The focus is thereby not on the side mirror as a monolithic system, but on its constituent modules, which are relevant from a development and control perspective and can be further developed independently. The presented example is deliberately simplified, as the primary objective of this work is to illustrate the MSPLE methodology rather than to provide a fully detailed product representation. Accordingly, the content is reduced and exemplified without claiming suitability for direct productive use.

The logical-functional modeling forms the foundation for analyzing the system. It enables the identification of functional relationships and dependencies, reveals functional chains, and highlights those system components that exhibit functional variability. The logical system structure, derived from the overall system architecture, serves as the reference framework for this modeling step.

Based on this structure, variability is explicitly captured in the Cameo models and formalized through the definition of variation points. These variation points are consistently documented and managed within a central feature model, which represents the key element for describing system variability. Initially, the feature model captures functional variability; as the modeling progresses toward lower levels of abstraction, it is further extended to include physical variability.

Within this framework, variability can be assigned either to individual system components or to groups of components forming modules. These modules represent logical aggregations of system elements and act as carriers of variability across the system. The precise definition of module boundaries is not the primary focus of this work; in general, a module may encompass one or more system components.

The methodology is applied consistently along the RFLP process using a multi-level ("Z-shaped") modeling approach [59]. In principle, all RFLP elements, requirements, functions, logical structures, and physical representations, are defined and refined across multiple system levels. However, within the scope of this case study, the focus is placed on selected aspects in order to illustrate the methodology in a clear and comprehensible manner.

At system level 1, all RFLP elements are defined at a high system level and serve as a reference for subsequent refinements. While, in general, these elements are further elaborated across all system levels, this work emphasizes the modeling steps that are most relevant for demonstrating variability and modularization. In particular, at system level 3, the logical components and modules of the side mirror are defined, together with their associated features. Building on this, system level 4 specifies these modules in greater detail and introduces their feature-dependent variants.

These modules are subsequently realized at the physical level through corresponding CAD models. The resulting physical variants are consistently linked back to the logical components, ensuring coherence across abstraction layers. At system level 4, module-specific variants are defined, which are then aggregated at system level 3 to form complete side mirror variants. These variants can be reused across different projects depending on the selected features, thereby enabling efficient configuration within a product line.

Although the overall approach encompasses a comprehensive multi-level RFLP modeling, the example deliberately focuses on selected steps. This allows for a clear demonstration of how variability is defined, structured, and propagated across abstraction levels, and how reusable, feature-based product variants can be derived.

To structure and explain the example, this section is divided into several subsections, also illustrated in Figure 13:

- Section 7.1 presents the relevant requirements and functional models, including exemplary links to the feature model. In particular, the requirements of the side mirror are linked to the feature model at system level 3 of the vision system.
- Section 7.2 introduces the logical structure, highlighting the modular composition of the side mirror as well as the associated features. This includes an exemplary mapping between logical components and feature models.
- Section 7.3 presents the physical structures and demonstrates how the selection of features and configurations directly influences the corresponding CAD models. The linkage between feature models and CAD artifacts is illustrated, along with the derivation of different system variants.

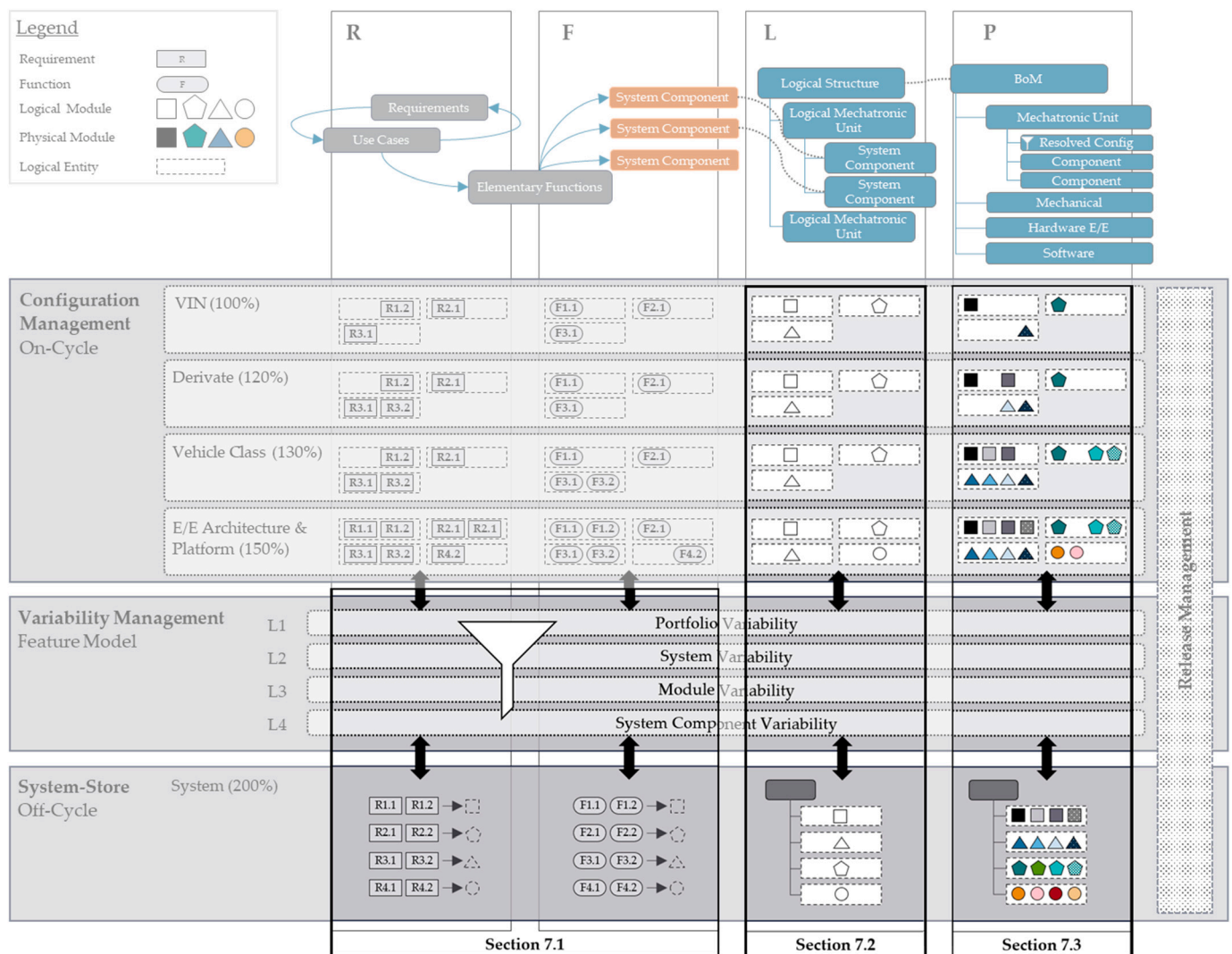


Figure 13. Structure of Section 7 and its content.

In addition, the case study demonstrates that, beyond the linkage of artifacts to features, direct relationships between R, F, L, and P elements are established. This ensures full traceability across all abstraction levels and enables consistent propagation of variability throughout the entire development process.

7.1. Requirements and Functional Modeling

This section presents the requirements and functional models developed within the side mirror case study, as well as their integration within the proposed MSPLE methodology. Although requirements and functional models are created across all system levels, the case study focuses exclusively on selected artifacts at system level 3 of the vision system. More specifically, only those elements relevant to the side mirror are discussed. The same principle is applied to the functional architecture, for which only representative excerpts of system level 4 are presented.

The modeling process begins on the left-hand side of the V-model with requirements engineering. For the purposes of this case study, the analysis starts directly at system level 3. Requirements originating from Codebeamer were integrated into the system structure of the vision system, as illustrated in Figure 14. The scope of the implementation was intentionally restricted to functional requirements, while non-functional requirements were excluded. Furthermore, particular attention is given throughout this section and the subsequent case study to the mirror heating and adjustment & folding mechanism features, which are shown at a higher level of detail.

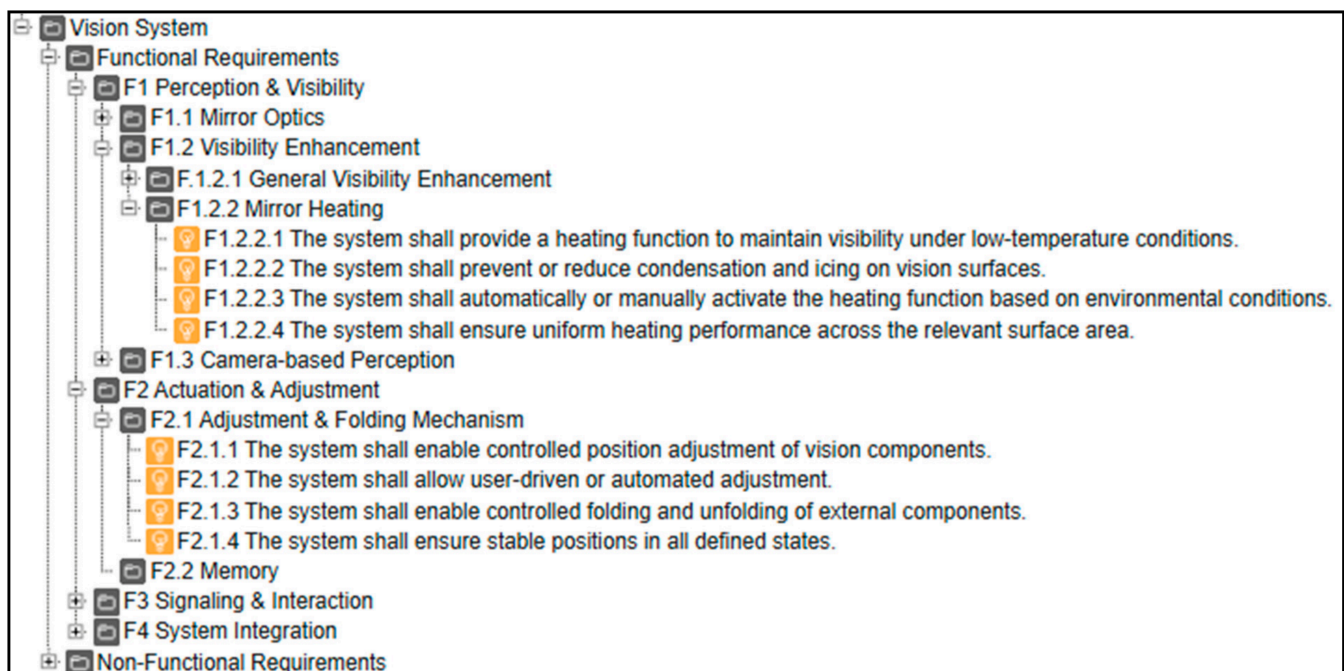


Figure 14. Requirements from Codebeamer, integrated into the system architecture of the vision system.

Since variability originates from requirements, the vision system feature model was derived from the variability represented by the modeled requirements and complemented by feature definitions that had already been established on higher system levels. The resulting feature model, shown in Figure 15, is located on system level 3 and establishes explicit traceability links between requirements and features. At its highest level of abstraction, the model consists of the five primary features adjustment, folding mechanism, mirror heating, signaling and interaction, and visual and perceptual function. These top-level features are subsequently refined into more detailed sub-features, providing a structured representation of system variability.

Vision System A

☐	Title	Type	Def...	Commercial	Variability Usage	Sequence Nu...	Action
1	☐ Adjustment	Variant		No	Mandatory ▼	1	▼
2	☐ Memory	Value				1	▼
3	☐ Mirror Adjustment manual	Value				1	▼
4	☐ Mirror Adjustment electrical	Value				1	▼
5	☐ Folding Mechanism	Variant		No	Mandatory ▼	1	▼
6	☐ Folding Mechanism manual	Value				1	▼
7	☐ Folding Mechanism electrical	Value				1	▼
8	☐ Mirror Heating	Variant		No	Mandatory ▼	1	▼
9	☐ no heating	Value				1	▼
10	☐ electrical heating	Value				1	▼
11	☐ Signaling and Interaction	Variant		No	Mandatory ▼	1	▼
12	☐ Lighting	Value				1	▼
13	☐ Warning Notification	Value				1	▼
14	☐ Projection Mode	Value				1	▼
15	☐ Visual and Perceptual Function	Variant		No	Mandatory ▼	1	▼
16	☐ Mirror Optics	Value				1	▼
17	☐ Vision Improvement	Value				1	▼
18	☐ Camera-based Perception	Value				1	▼

Figure 15. Feature model at system level 3 depicting the vision system.

To establish the functional architecture on system level 4, it is necessary to introduce the module-specific feature models defined on system level 3 and populated with detailed content on system level 4. Consequently, the logical components defined on system level 3 serve as the foundation for constructing the functional architecture on the subsequent system level. The functional architectures of system level 3 and higher abstraction levels are outside the scope of the presented case study and are therefore not discussed further.

Although the module-specific feature models are described in detail in Section 7.2, two representative examples, the side mirror glass and side mirror housing modules, are shown in Figure 16. Within these module-specific feature models, module variability is represented entirely through features and their associated variants. The mirror heating feature is defined within the feature model of the side mirror glass module, whereas the folding mechanism feature is represented within the feature model of the side mirror housing module. Since the realization of these feature variants requires corresponding system functionality, the associated functions are modeled in Cameo Systems Modeler and become integral elements of the functional architecture.

Side Mirror Glass A

☐	Title	Type	Def...	Commercial	Variability Usage
1	☐ Mirror Shading	Variant		No	Mandatory ▼
2	☐ Mirror Blindspot	Variant		No	Mandatory ▼
3	☐ Mirror Heating	Variant		No	Mandatory ▼
4	☐ no heating	Value			
5	☐ electric heating	Value			

Side Mirror Housing A

☐	Title	Type	Def...	Commercial	Variability Usage
1	☐ Steering	Variant		No	Optional ▼
2	☐ Color	Variant		No	Optional ▼
3	☐ Integrated Turn Lights	Variant		No	Mandatory ▼
4	☐ Folding Mechanism	Variant		No	Mandatory ▼
5	☐ manual	Value			
6	☐ electrical	Value			

Figure 16. Module-specific feature models of the side mirror (side mirror glass and side mirror housing).

Figure 17 illustrates the import of the previously introduced module-specific feature models into Cameo Systems Modeler. The figure shows the modules side mirror glass

and side mirror housing together with their associated configuration rules governing module variability. For the side mirror glass module, the three variants high, mid, and low, previously modeled in 3DEXPERIENCE, are imported. Similarly, the side mirror housing module comprises the two variants basic and comfort. The imported module-specific feature models are linked to the functional architecture through explicitly defined variation points within the functional model. This mechanism establishes a direct connection between variability modeling and functional realization, ensuring that functional behavior can be configured consistently according to the selected module variants.

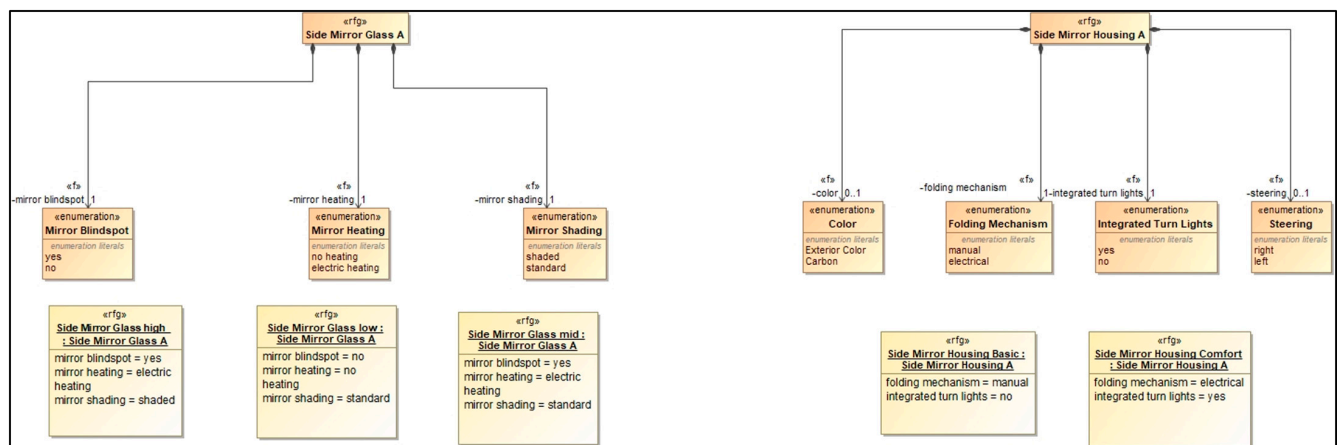


Figure 17. Import of module-specific feature models into Cameo Systems Modeler.

A representative excerpt of the functional architecture developed in Cameo Systems Modeler, which has been referenced throughout this section as the central representation of the system's functional behavior and variability realization, is shown in Figure 18.

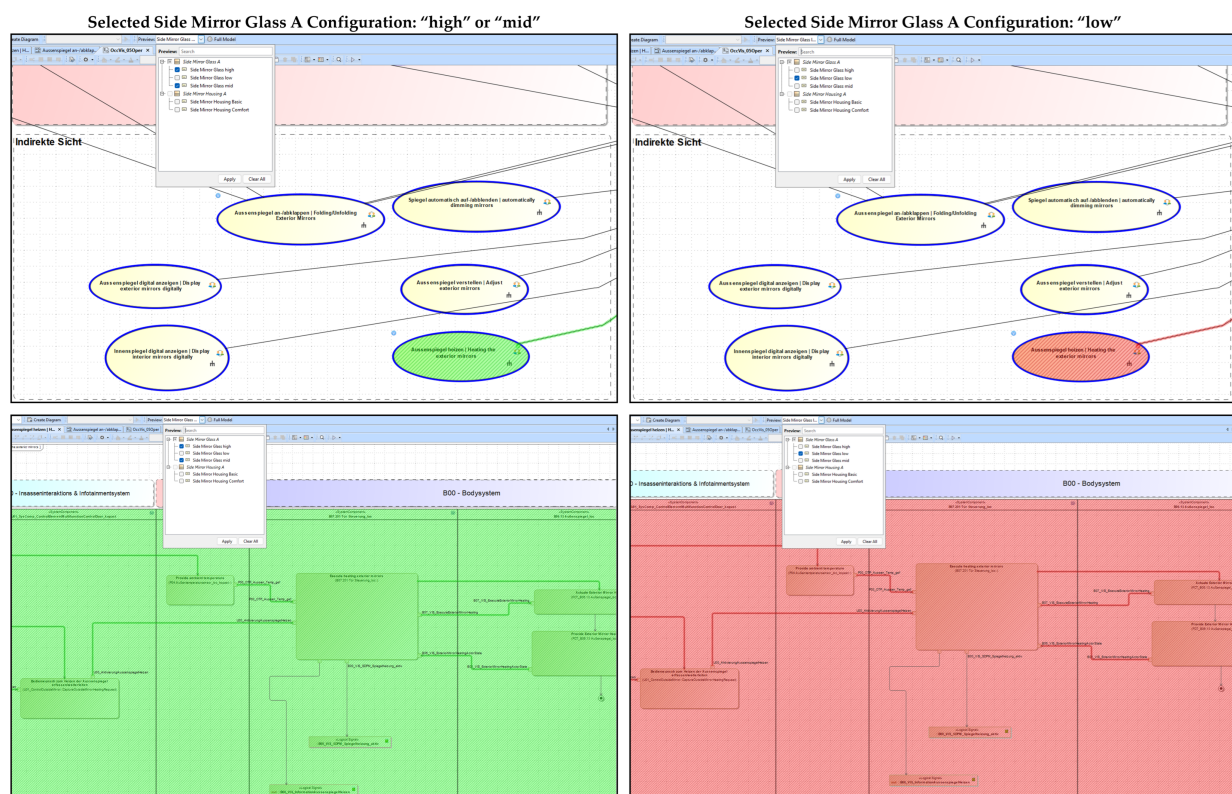


Figure 18. Functional architecture of the mirror heating use case.

The upper part of the figure presents a set of modeled use cases, including the mirror heating use case. The lower part illustrates its functional realization within the system architecture. By selecting specific variants of the side mirror glass module, the model dynamically determines whether the mirror heating functionality is configured as no heating (highlighted in red) or electric heating (highlighted in green). This configuration behavior demonstrates the direct relationship between module variability and functional realization: the electric heating functionality is available only for the high and mid variants of the side mirror glass module, whereas it is not included in the low variant. Through this mechanism, traceability is maintained from requirements to features, from features to module variants, and ultimately to the functions realizing the selected configuration.

7.2. Logical Architecture and Modular Structuring

Based on the requirements and functional models described in Section 7.1, the logical architecture of the side mirror is derived as the next step of the RFLP process. This step translates the previously defined functional relationships into a structured system representation and establishes the foundation for modularization.

For the definition of the logical structure, the LRSA is used as the overarching framework. The LRSA represents a cross-brand, project-independent system hierarchy within the Volkswagen Group and provides a standardized structure for organizing logical system elements. Figure 19 illustrates this structure and shows how the side mirror is embedded within the higher-level system context of the vehicle. In particular, the side mirror is located within the vision system and is positioned as a subsystem within the broader logical solution structure.

Within this framework, the side mirror is decomposed into its constituent logical modules. As shown in Figure 19, these modules include the mirror housing, the mirror glass, the adjustment mechanism, and the mirror camera. These modules are not defined at the highest system level but emerge through the refinement of the system along the RFLP “Z-shaped” modeling approach, specifically at system level 3. At this level, logical components are defined and aggregated into modules that represent coherent, functionally and structurally consistent units.

Each of these modules represents a carrier of functionality and variability and can be developed and managed independently to a certain extent. At the same time, they remain embedded within the overall system architecture defined by the LRSA, ensuring consistency across system levels and domains.

In addition to their structural definition, the modules are closely linked to variability modeling. Each module is associated with its own feature model, capturing the module-specific variability. Figure 20 presents exemplary feature models for the modules of the side mirror, such as the side mirror glass, side mirror housing, side mirror camera, and side mirror adjustment mechanism.

These feature models define the possible variants of each module, for instance in terms of functionality (e.g., heating and blind spot detection), design characteristics (e.g., color), or technical realizations.

The variability captured in these module-specific feature models originates from two sources. First, it is derived from higher-level feature models, particularly from the feature model defined at system level 3 of the vision system. This ensures that module variability is consistent with system-level variability and supports top-down traceability. Second, additional module-specific features may be introduced to capture variability that is only relevant at the respective module level. This combination enables both alignment with the global system context and sufficient flexibility at the module level.

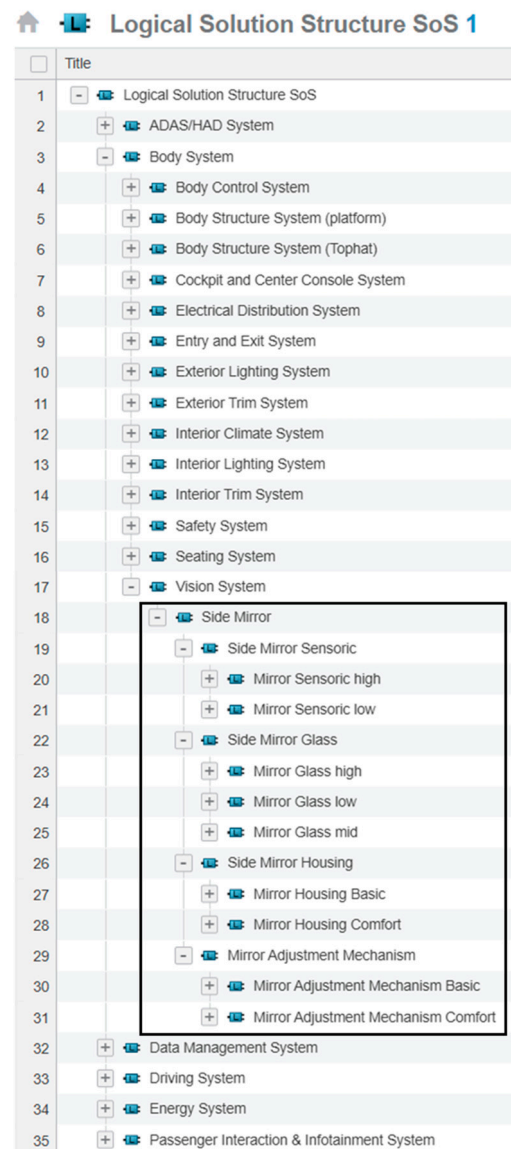


Figure 19. Representation of the logical structure using the side mirror as an example.

To ensure overall system consistency, the individual module feature models are not isolated but are linked to a central, higher-level feature model, called enterprise portfolio model. This global feature model contains overarching features such as brands or markets. Dependencies and constraints between module-level features and global features are explicitly defined, enabling a consistent configuration logic across system levels.

Through this integration, the logical architecture and the feature modeling layer are tightly coupled. The logical modules define the structural decomposition of the system, while the feature models describe the variability of these modules. The linkage between both layers enables the systematic control of modules based on feature selections and forms the basis for subsequent variant derivation.

Overall, this section demonstrates how the logical architecture serves as the central layer for combining system structuring and variability modeling. By embedding the side mirror within the LRSA, defining modular and logical structures at system level 3, and linking these modules to feature models, a consistent and scalable foundation for MPD and SPLE is established. This structure is subsequently used in Section 7.3 to derive physical realizations and corresponding CAD-based variants.

Side Mirror Housing A						
	Title	Type	Def...	Commercial	Variability	Action
1	Steering	Variant		No	Optional	▼
2	right	Value				▼
3	left	Value				▼
4	Color	Variant		No	Optional	▼
5	Exterior Color	Value				▼
6	Carbon	Value				▼
7	Integrated Turn Lights	Variant		No	Mandatory	▼
8	yes	Value				▼
9	no	Value				▼
10	Folding Mechanism	Variant		No	Mandatory	▼
11	manual	Value				▼
12	electrical	Value				▼

Side Mirror Sensoric A						
	Title	Type	Def...	Commercial	Variability	Action
1	Camera Type	Variant		No	Mandatory	▼
2	360-degree Camera	Value				▼
3	Sideview Camera	Value				▼
4	no Camera	Value				▼
5	Camera Resolution	Variant		No	Optional	▼
6	HD Camera	Value				▼
7	Full-HD Camera	Value				▼
8	Sensoric	Variant		No	Mandatory	▼
9	PDC	Value				▼
10	no PDC	Value				▼

Side Mirror Glass A						
	Title	Type	Def...	Commercial	Variability	Action
1	Mirror Shading	Variant		No	Mandatory	▼
2	shaded	Value				▼
3	standard	Value				▼
4	Mirror Blindspot	Variant		No	Mandatory	▼
5	yes	Value				▼
6	no	Value				▼
7	Mirror Heating	Variant		No	Mandatory	▼
8	no heating	Value				▼
9	electric heating	Value				▼

Mirror Adjustment Mechanism A						
	Title	Type	Def...	Commercial	Variability	Action
1	Adjustment Type	Variant		No	Mandatory	▼
2	electrical	Value				▼
3	manual	Value				▼

Figure 20. Module-specific feature models of the side mirror illustrating component-level variability.

7.3. Physical Models and Variant Derivation Through Feature-Based Configuration

The development of physical system models is based on the structured transformation of logical system definitions into concrete realization artifacts. This transformation represents the transition from the logical layer (L) to the physical layer (P) within the RFLP framework, where abstract system structures are instantiated as physical system components.

The physical realization builds upon the modular system structure defined within the logical architecture. At the logical layer, the system is decomposed into modules and associated with variability, thereby establishing both the structural decomposition and the variability backbone of the system. This modularization provides a consistent foundation for the derivation of physical representations and ensures that system elements can be reused and configured across different product instances.

As a result, the logical architecture serves as the authoritative reference for the physical realization. It defines not only the system structure but also the relationships between modules and their variability, which directly guide the instantiation of physical system elements. This enables a systematic linkage between variability definitions and system components, ensuring that configuration decisions made at the logical layer are consistently transferred to the physical layer and reflected in the resulting system realizations.

The example demonstrates how variability definitions are transformed into product structures and concrete engineering artifacts within a product lifecycle management (PLM) environment, using the 3DEXPERIENCE platform. Figure 21 provides an overview of the modular decomposition of the side mirror system together with its corresponding product structure. It illustrates how the defined modularization is reflected in the physical system representation and forms the basis for the subsequent description of variability realization and configuration.

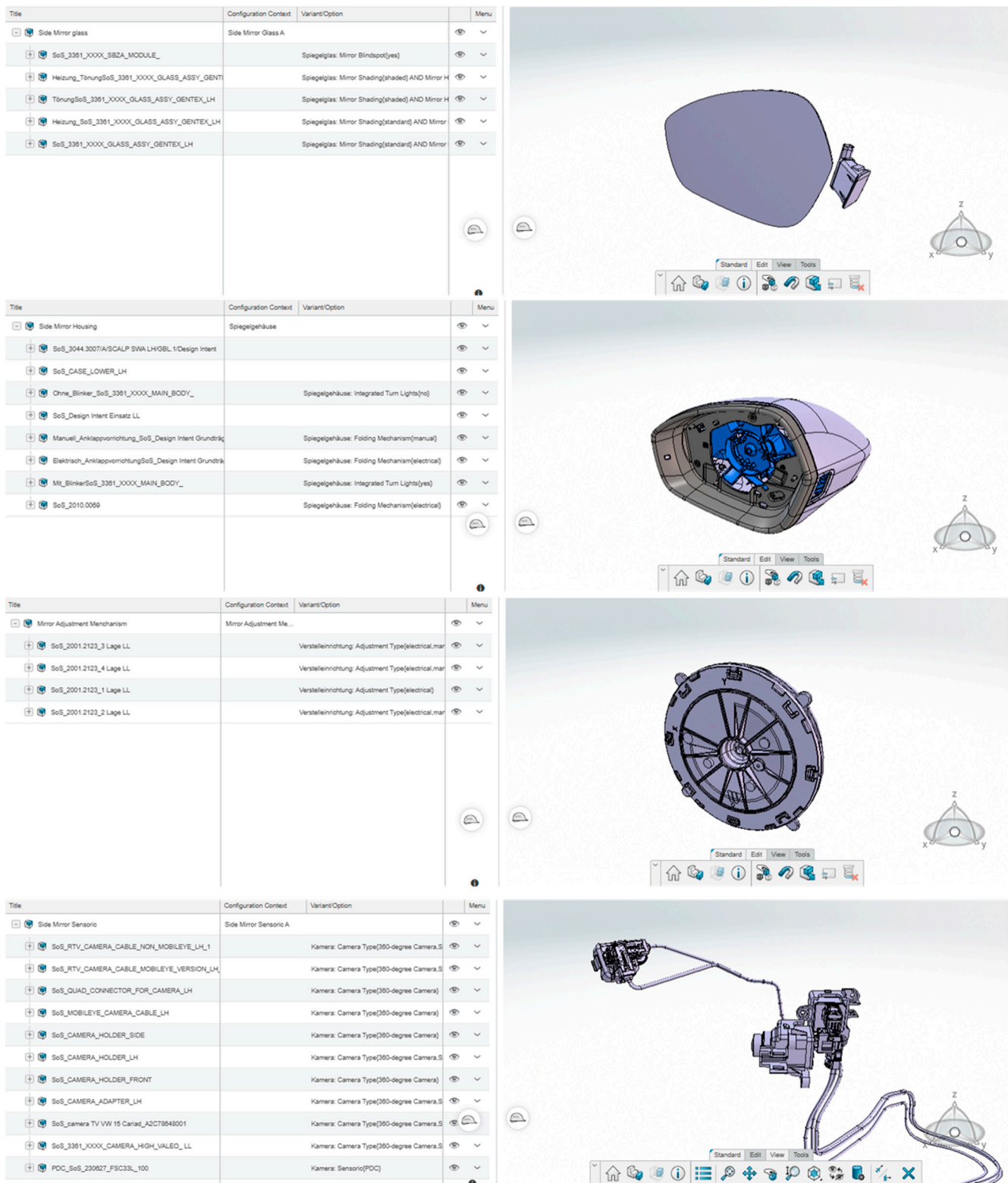


Figure 21. Product structure of the modules of the side mirror system.

Variability is defined and governed within the variability model of the problem space through feature models. These models capture module-specific characteristics, dependencies, and constraints, and are linked across hierarchical system levels. Configuration rules derived from these models define valid feature combinations and enable the systematic derivation of module-specific configurations. As a result, variability defined at the feature level is consistently transferred into the solution space. Non-functional and invariant

requirements are not part of the variability model but are incorporated into the architecture when they are relevant to system behavior.

The linkage to the physical layer is established by assigning a configuration context to the product structure, thereby enabling the consistent propagation of variability into physical system representations. This is realized through a formal mapping between features and product structure elements, for example by means of allocation matrices. As a result, feature selections drive the activation and instantiation of physical components within a 150% product structure. In this way, feature-based variability is consistently propagated into the physical realization, thereby enabling the systematic derivation of valid product variants. Figure 22 illustrates this allocation process using the side mirror housing as an example, highlighting how feature selections govern the inclusion and configuration of corresponding physical components within the product structure.

☐	Instance	Integrated Turn Lights		Folding Mechanism	
		yes	no	manual	electrical
1	SoS_3044.3007/A/SCALP SWA LH/GBL.1/Design Intent.1	☐	☐	☐	☐
2	SoS_CASE_LOWER_LH.1	☐	☐	☐	☐
3	SoS_3361_XXXX_MAIN_BODY_1	☐	✓	☐	☐
4	SoS_Design Intent Einsatz LL.1	☐	☐	☐	☐
5	SoS_Design Intent Grundträger LL.1	☐	☐	✓	☐
6	Elektrisch_AnklappvorrichtungSoS_Design Intent Grundträger LL.1	☐	☐	☐	✓
7	Mit_BlinkerSoS_3361_XXXX_MAIN_BODY_1	✓	☐	☐	☐
8	SoS_2010.0069.1	☐	☐	☐	✓

Figure 22. Definition of constraints for the side mirror housing.

The linkage between the logical (L) and physical (P) layers is realized through the modular system structure and the systematic allocation of logical elements to corresponding physical representations. The logical architecture defines what is to be realized and structures the variability of the system, whereas the physical layer specifies how this variability is implemented in terms of product structures and concrete realization artifacts.

A key outcome of the physical layer is the generation of CAD-based realization artifacts that represent the geometric and structural implementation of system variants. These models are maintained as 150% representations, encompassing all components required to cover the complete variability space. Within this context, CAD models constitute realization artifacts of the SPLE solution space, while the product structure reflects the modular decomposition defined in the logical architecture. The mapping between features and components ensures that variability is consistently instantiated at the physical level.

The configuration process subsequently resolves the 150% structure into valid product instances through the application of feature selections and configuration rules. This results in concrete system variants, such as different side mirror configurations, in which variations in physical components directly correspond to feature-level variability. Although this realization is exemplified using CAD models, the underlying principle is equally applicable to other domains, such as software, where feature selections govern the inclusion and configuration of components.

Figure 23 shows the 150% product structure with its configuration context (left-hand side), where feature-level variability is mapped to product structure elements. The right-hand side illustrates resolved configurations (Mirrox A high, Mirrox B mid and Mirrox C low), derived from feature selections and rules, resulting in distinct physical variants. Differences in functional features directly reflect these selections, demonstrating consistent variability propagation and end-to-end traceability.

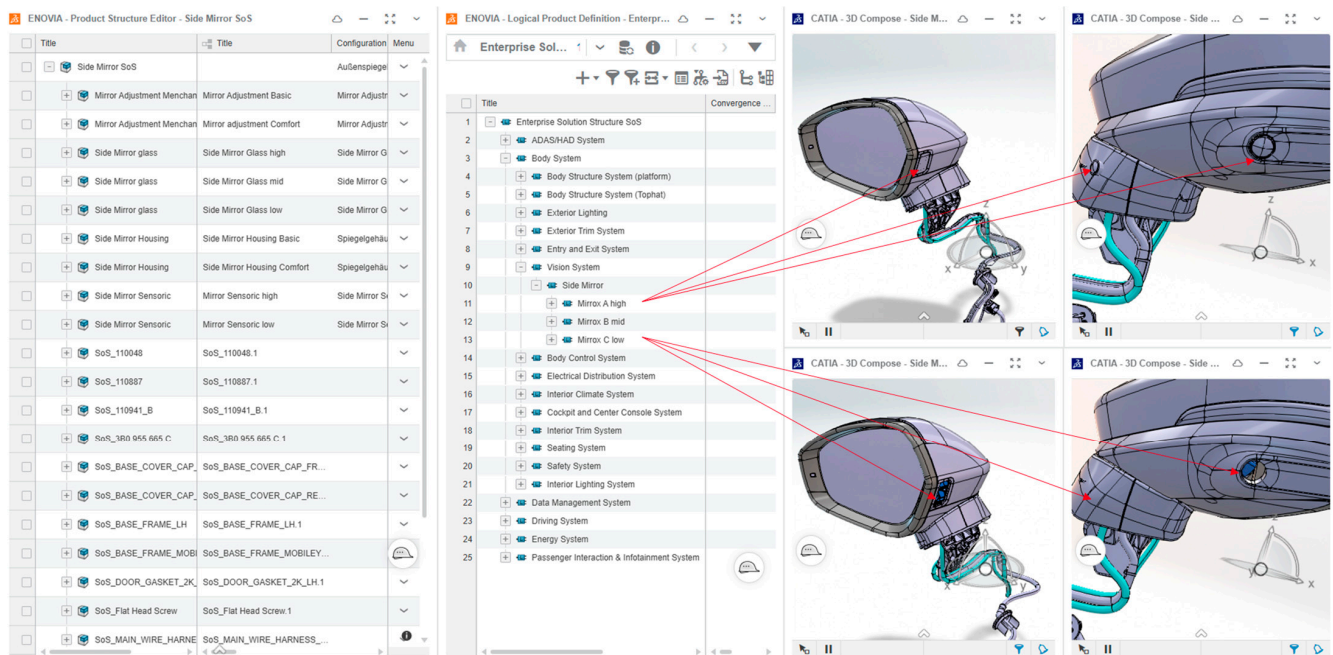


Figure 23. Side mirror structure and logical structure with resolved system configurations.

The proposed MSPLE method enables modeling of arbitrarily nested systems following a SoS paradigm. At each hierarchical level, configurations from level $(n - 1)$ are integrated into level (n) via configuration contexts and constraints, resulting in recursively composed and configurable product structures with consistent propagation of variability and constraints.

Resolved configurations can be reused as predefined configuration instances, representing validated solution elements within the system architecture. These SoS configurations are integrated into the enterprise solution structure and can be reused at the vehicle level by incorporating them into the respective product structure, enabling a seamless transition from system configuration to vehicle realization with ensured consistency, reuse, and controlled variability across the lifecycle.

Through this MSPLE methodology, variability is consistently propagated across the RFLP layers. The logical architecture defines the system structure and variability, while the physical layer realizes these concepts in concrete system representations.

8. Evaluation

This chapter evaluates the MSPLE methodology introduced in Section 6 based on the side mirror example presented in Section 7. Section 8.1 presents the actual evaluation of the methodology. Subsequently, Section 8.2 addresses the research questions, discusses the results, and outlines the limitations of the proposed methodology.

As already mentioned in Section 4, the evaluation follows the DRM approach shown in Figure 4. The assessment is therefore based on a comparison between the requirements defined in Section 5 and the MSPLE methodology proposed in Section 6. At first glance, such an evaluation approach may appear subjective or insufficiently rigorous. However, the claims made during the evaluation are substantiated through the exemplary instantiation of the MSPLE methodology using the side mirror use case presented in Section 7. Based on this demonstrative implementation, it is argued that the chosen form of evaluation is appropriate within the scope of this research.

For this purpose, the requirements summarized in Table 2 are used together with the same evaluation matrix previously applied for assessing existing SPLE and MPD ap-

proaches. In addition, the evaluation critically reflects upon the obtained results and is therefore complemented by a broader discussion of the MSPLE methodology, its implications, and its limitations.

8.1. Evaluation of the Proposed MSPLE Methodology

Table 4 presents the evaluation of the proposed MSPLE methodology. The results indicate that the majority of the identified requirements are comprehensively addressed, while also revealing that certain aspects still leave room for further improvement. To further improve the transparency of the evaluation, explicit assessment criteria are defined. A requirement is classified as fulfilled if (1) the MSPLE methodology provides the necessary methodological concepts and (2) the capability is demonstrated within the side mirror case study. A requirement is classified as partially fulfilled if the requirement is conceptually addressed but lacks either methodological guidance or sufficient case-study evidence. A requirement is classified as not fulfilled if the requirement lies outside the scope of MSPLE and no methodological support is provided. Furthermore, a detailed mapping between the artifacts and methodological elements described in Section 6, their instantiation within the case study presented in Section 7, and the resulting assessment in Table 4 is provided in Appendix A (Table A1).

Table 4. Evaluation of the proposed MSPLE methodology.

ID	Requirement	MSPLE	SPLE	MPD
R-01	The methodology shall represent mechatronic, hardware, or software systems.	●	●	●
R-02	The methodology shall support the representation of RFLP layers.	●	●	●
R-03	The modules shall be uniquely and unambiguously describable.	●	●	○
R-04	The modules shall rely on a consistent description methodology.	●	○	●
R-05	The modules shall be combinable within a system architecture.	●	●	●
R-06	The modules shall support the representation of relationships to other modules.	●	●	●
R-07	The modules shall be describable through defined properties and characteristics.	●	●	○
R-08	The modules shall provide clearly defined interfaces.	○	○	●
R-09	The modules shall be configurable based on system requirements.	●	●	○
R-10	The modules shall be representable in multiple variants.	●	●	○

●: Requirement satisfied; ●: Requirement partially satisfied; ○: Requirement not satisfied.

The following sections provide a detailed discussion of these assessments by explaining, for each individual requirement, the rationale behind the respective evaluation.

The evaluation first focuses on the requirements that are considered to be fully satisfied by the proposed MSPLE methodology.

Requirement R-01, which concerns the representation of mechatronic, hardware, and software systems, can be regarded as fulfilled. The MSPLE methodology is inherently domain independent, as it links all artifacts to the physical layer and the modular system structure. In the presented case study, this linkage has been demonstrated using hardware artifacts, specifically CAD models of the side mirror. However, the same structural principle can be applied equivalently to other domains, such as software artifacts or parameter sets. Consequently, the MSPLE methodology does not impose any restrictions on the

type of realization and is therefore capable of representing systems across domains in a consistent manner.

Similarly, R-02 is fully addressed through the explicit realization of the RFLP framework. The MSPLE methodology demonstrates a continuous and consistent representation across requirements, functional, logical, and physical layers. This end-to-end traceability has been illustrated throughout the case study and ensures that all relevant system perspectives are systematically integrated.

With regard to module definition and description, the MSPLE methodology also fulfills several key requirements. It has been shown how modules can be uniquely and unambiguously described (R-03) by combining logical structuring with feature-based characterization. Furthermore, the MSPLE methodology demonstrates how modules are systematically assigned to the functional and logical layers and integrated into the overall architecture (R-05). This ensures that modules are not only structurally defined but also consistently embedded within the system context.

In addition, modules are described through explicitly defined properties and characteristics by linking them to feature models, thereby fulfilling R-07. This attribute-based description enables a precise specification of module variability and provides the foundation for configuration and reuse.

The representation of relationships between modules (R-06) is also fully addressed. The MSPLE methodology shows how dependencies and interactions between modules can be captured both within the logical architecture and through feature-based constraints. This is exemplified in Section 7.3, where module interdependencies are explicitly modeled and linked to variability mechanisms.

Moreover, the MSPLE methodology demonstrates how modules can be configured based on system requirements (R-09). By linking features to requirements and using them as decision variables, it becomes possible to systematically control the selection and instantiation of modules. This configuration capability is consistently applied across system levels, enabling a structured derivation of system variants.

Additionally, the MSPLE methodology clearly supports the representation of modules in multiple variants (R-10). This is illustrated in the side mirror example, where different module variants, such as alternative realizations of the side mirror housing, are instantiated and reused across different side mirror configurations. The reuse of submodule variants in different higher-level system variants demonstrates the scalability and flexibility of the MSPLE methodology.

However, the evaluation also revealed requirements that are only partially fulfilled or not sufficiently addressed by the proposed MSPLE methodology.

Requirement R-04, which concerns the existence of a consistent description methodology for modules, is assessed as only partially fulfilled. While the MSPLE methodology demonstrates how modules can be described and characterized using feature-based approaches, it does not provide explicit guidance on how modules should be defined or partitioned. In particular, no systematic criteria or methodological guidelines are provided for determining the appropriate level of granularity or for defining module boundaries. Although the LRSA is used as a structural orientation, it does not constitute a prescriptive method for modular decomposition. As a result, the definition and granularity of modules remain context dependent and must be determined individually for each system. This limitation highlights an important area for future research, particularly with regard to developing systematic approaches for module partitioning and modular architecture design.

A similar limitation can be observed for requirement R-08, which addresses the definition of clearly specified interfaces. Within the industrial context, the reuse of modules across platforms and vehicle projects requires the existence of standardized and pre-defined

interfaces to ensure compatibility and integrability. However, this aspect is not explicitly addressed in the presented MSPLE methodology or within this publication as a whole. While the MSPLE methodology demonstrates how features can be used to define the conditions under which modules can be reused, it does not specify how technical interfaces between modules should be designed, described, or governed. Due to the scope and simplification of the presented use case, it was not possible to elaborate on the nature and structure of such interfaces.

This limitation is intentional and reflects the scope of the MSPLE methodology. MSPLE focuses on the governance of variability and modular architectures rather than on the specification of technical interfaces themselves. Interface definitions, compatibility rules, and interface evolution mechanisms are highly dependent on the specific system, architecture, technologies, and engineering domains involved. Consequently, a generic interface specification approach cannot be prescribed within the proposed methodology. Nevertheless, interface governance remains a prerequisite for the successful industrial adoption of MSPLE and should therefore be addressed by complementary engineering approaches.

Despite these limitations, it can be concluded that the majority of the requirements are fulfilled. The proposed MSPLE methodology thereby establishes a coherent framework that enables the application of variability management principles for the systematic control of modules and artifacts, as well as their reuse across different system contexts. By combining feature-based variability modeling with modular system structuring along the RFLP framework, the MSPLE methodology provides a consistent foundation for managing complex, variant-rich systems throughout the development lifecycle, particularly in alignment with the principles of the V-model.

8.2. Overall Assessment and Discussion

Based on the evaluation results, the research questions defined in Section 4 can be addressed coherently. The requirements derived in Section 5 are largely validated through the instantiation and subsequent evaluation of the proposed MSPLE methodology. The findings clearly indicate that neither SPLE nor MPD, when applied in isolation, is capable of comprehensively fulfilling the identified requirements. In particular, the lack of integration between variability management and modular system structuring emerges as a fundamental limitation of existing approaches. While SPLE provides strong mechanisms for handling variability and configuration, it lacks an explicit integration with modular architectures. Conversely, MPD offers well-established principles for system decomposition but does not sufficiently address systematic variability control. This structural imbalance underlines the necessity of an integrated methodology.

The proposed MSPLE methodology provides an initial answer to the second research question by demonstrating, through a proof-of-concept case study, how SPLE principles and MPD concepts can be combined within a unified framework. By integrating feature models with modular architectures and embedding them within the RFLP structure, the MSPLE methodology demonstrates the potential to support a consistent representation and control of variability across multiple system levels. Further validation is required to assess the scalability of the approach in larger industrial environments. This integration allows both the structural definition of modules and their dynamic configuration based on system requirements, thereby bridging the gap between architecture design and variability management. At the same time, the evaluation indicates that the MSPLE methodology should be understood as a conceptual and structural framework rather than a fully operationalized engineering solution.

Despite its strengths, several limitations of the proposed MSPLE methodology must be acknowledged. First, the validation is currently based on a simplified use case, namely

the side mirror example. Consequently, the presented case study demonstrates the feasibility and internal consistency of the methodology rather than providing evidence for lifecycle-wide applicability. While this example effectively demonstrates the feasibility and internal consistency of the approach, it does not fully capture the scale, heterogeneity, and complexity of real-world automotive systems. Consequently, further validation in more complex and large-scale industrial scenarios is required to assess the robustness and scalability of the MSPLE methodology.

Second, the proposed methodology does not explicitly address interface specification and interface lifecycle management. While interface definitions are essential for enabling module interoperability and reuse, their concrete realization depends strongly on the individual system context. Therefore, MSPLE currently assumes the existence of appropriately defined interfaces but does not provide a methodology for deriving or governing them. Future research should investigate how interface governance, compatibility management, and interface evolution can be integrated with the proposed framework without compromising its domain-independent character.

Third, the operationalization of the MSPLE methodology remains limited. While the conceptual integration of RFLP and feature-based variability modeling is clearly articulated, practical aspects such as standardized modeling guidelines and integration into existing engineering processes are not yet fully elaborated. This limits the immediate applicability of the MSPLE methodology in industrial contexts and highlights the need for further development toward a fully deployable engineering approach.

Another limitation of the proposed methodology concerns tool interoperability and engineering process automation. The MSPLE does not prescribe how information should be exchanged between the various engineering tools involved. In industrial environments, tool integration is often realized through dedicated interfaces, data transformations, or organization-specific integration solutions. Consequently, the practical implementation of MSPLE still requires additional integration mechanisms at the tooling level. From a future perspective, SysML v2 appears to be a promising enabler for addressing this challenge. Its standardized and machine-readable model representation could provide a common integration layer across engineering disciplines and tools, thereby reducing the need for project-specific interfaces. Furthermore, the textual nature of SysML v2 creates opportunities for advanced automation concepts, including AI- and agent-supported engineering workflows capable of interpreting, analyzing, and maintaining engineering artifacts across multiple domains. However, the integration of such mechanisms is outside the scope of the present work and represents an important direction for future research.

Finally, the effectiveness of the MSPLE methodology strongly depends on the quality and consistency of the underlying feature models, as well as on their linkage to the corresponding artifacts and the early integration of development artifacts such as requirements. Inaccurate, incomplete, or inconsistently maintained variability models can negatively affect configuration correctness and overall system consistency. Consequently, robust governance mechanisms and methodological support for feature model management are essential for the successful application of the proposed MSPLE methodology.

9. Threats to Validity

This study is subject to several limitations that should be considered when interpreting the results and assessing the applicability of the proposed MSPLE methodology. Following established research practices, potential threats are discussed with respect to construct validity, internal validity, external validity, and conclusion validity [61].

A potential threat to construct validity arises from the derivation of requirements used for developing the proposed MSPLE methodology. The requirements were obtained

through an industrial application analysis of the MPD logic employed within the Volkswagen Group and complemented by findings from the literature on SPLE and product modularization. Although this combination provides both practical relevance and theoretical grounding, the identified requirements are inevitably influenced by the specific characteristics, processes, and organizational structures of the investigated industrial environment. Consequently, certain requirements may reflect company-specific assumptions that are not universally applicable across different organizations or development contexts.

Threats to internal validity primarily originate from the design-oriented nature of the research approach. The proposed MSPLE methodology was developed through the integration and interpretation of existing concepts from SPLE and MPD. As with any synthesis-oriented approach, decisions regarding abstraction levels, selection of concepts, and relationships between model elements involve a degree of researcher interpretation. While these decisions were systematically guided by previously defined requirements and grounded in existing theory, alternative interpretations or design choices may lead to different methodological solutions.

With respect to external validity, the generalizability of the findings is inherently limited by the empirical basis of the study. The industrial analysis was conducted within a single OEM environment in the automotive domain. Although the identified challenges, such as increasing variability, cross-domain dependencies, and architectural complexity, are representative of broader trends in SDVs, the proposed MSPLE methodology has not yet been validated across multiple organizations or industrial contexts. Furthermore, existing modularization approaches have historically been developed primarily for hardware-oriented systems, while the presented work addresses highly integrated mechatronic systems consisting of hardware, E/E, and software elements. Additional evaluations in different industrial settings and across different system architectures are therefore required to assess broader applicability.

Threats to conclusion validity arise from the current evaluation strategy. The proposed MSPLE methodology was primarily assessed through a qualitative comparison against previously defined requirements. While this approach enables an initial assessment of practical relevance and methodological completeness, it does not provide quantitative evidence regarding measurable effects such as reductions in development effort, complexity, or lifecycle costs.

Despite these limitations, the selected research design provides a structured and theory driven foundation for the development of the proposed MSPLE methodology. The combination of established theoretical concepts with an industrial application context increases practical relevance while maintaining methodological rigor. Future work should therefore focus on broader industrial validation, quantitative assessment of the proposed MSPLE methodology, and evaluation across multiple organizations and system domains to strengthen the generalizability and empirical robustness of the findings.

10. Conclusions

In conclusion, this paper investigated the integration of variability management and product modularization within a model-based engineering context. The analysis demonstrates that existing approaches, including SPLE and MPD, do not sufficiently meet the requirements, as they lack a unified integration of variability and modular structuring. To address this gap, the MSPLE methodology is introduced as a conceptual framework, providing a foundation for future operationalization and industrial application.

Based on these findings, this paper makes two central contributions. First, it proposes an integrated approach that combines SPLE with MPD. This is achieved through feature model-based module governance and a formalized methodology. Second, it establishes

a consistent traceability framework along the V-model based on model-based SPLE and the RFLP paradigm. The presented case study demonstrates end-to-end traceability from requirements to physical realization within the scope of the implemented proof-of-concept. Building on this foundation, MSPLE establishes a conceptual methodological framework intended to govern variable modules through a model-based approach across the RFLP layers.

In relation to the V-model, MSPLE primarily focuses on the left-hand side, while validation and verification activities on the right-hand side are not yet explicitly addressed. However, extending the approach to the validation domain is a possible future research direction as it enables the continuation of the RFLP-based traceability chain towards test cases and validation artifacts. This ensures that variability is consistently propagated into test design and execution, enabling systematic, configuration-driven validation. The MSPLE methodology provides an important foundation for operationalizing the GSS. However, the definition of modules as well as the decomposition of the product into systems and subsystems are not addressed within the scope of this methodology. Instead, these elements must be specified by the respective company based on its own organizational, architectural, and process-related context.

For the implementation of the GSS, release management is required in addition to feature-based system governance. The evolution of development artifacts and variability over time introduces a temporal dimension that must be systematically managed to ensure consistent variability evolution. While MSPLE enables structured variability, it does not address the temporal dimension of variability, limiting consistent variability evolution across the lifecycle. In addition to system governance, further technical management disciplines, including change and release management, are also essential to support comprehensive lifecycle control.

The approach is subject to several limitations, including validation based on a simplified use case, the absence of explicit interface modeling and governance, and its currently limited industrial applicability. While the side mirror case study demonstrates the feasibility of the proposed approach and the integration of variability management, modular architectures, and RFLP-based traceability, broader validation in more complex automotive systems and across multiple industrial contexts remains future work.

Furthermore, its effectiveness depends on the quality and consistency of the underlying feature models and their governance. Future research should therefore focus on scalability, interface refinement, and methodological standardization to enable robust industrial application.

Despite these limitations, MSPLE provides a promising foundation for integrating MPD and SPLE and for exploring variability management as a model-based capability across the V-model. Additional industrial validation is required to assess its applicability across different system domains, organizations, and development lifecycles.

Author Contributions: Conceptualization, F.G., Y.L. and R.v.E.; methodology, F.G., Y.L. and R.v.E.; validation, F.G., Y.L., R.v.E., J.O. and T.V.; investigation, F.G., Y.L. and R.v.E.; resources, J.O. and T.V.; data curation, F.G., Y.L. and R.v.E.; writing—original draft preparation, F.G., Y.L. and R.v.E.; writing—review and editing, F.G., Y.L., R.v.E., J.O. and T.V.; visualization, F.G., Y.L. and R.v.E.; supervision, J.O. and T.V.; project administration, J.O. and T.V.; funding acquisition, T.V. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Volkswagen AG. We acknowledge support by the Open Access Publication Funds of Technische Universität Braunschweig.

Data Availability Statement: The datasets presented in this article are not readily available because of confidentiality and legal reasons. Requests to access the datasets should be directed to Richard von Eisebeck.

Acknowledgments: During the preparation of this manuscript, the authors Fabian Goihl, Yannick Lindebauer and Richard von Eisebeck used Microsoft Copilot (V. 2.20260602.20.0) for the purposes of improving readability and language. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Yannick Lindebauer reports financial support was provided by Volkswagen AG. Thomas Vietor reports financial support was provided by Volkswagen AG. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Disclaimer: The results, opinions, and conclusions of this paper are not necessarily those of Volkswagen AG.

Abbreviations

The following abbreviations are used in this manuscript:

BOM	bill of materials
CAD	computer-aided design
DRM	design research methodology
E/E	electrical/electronic
ECU	electronic control unit
FODA	feature-oriented domain analysis
GSS	group systems store
LRSA	logical reference system architecture
MBSE	model-based systems engineering
MEB	modular electric drive matrix
MPD	modular product development
MSPLE	modular systems product line engineering
MQB	modular transverse matrix
OEM	original equipment manufacturer
OVM	orthogonal variability model
PEP	product emergence process
PLM	product lifecycle management
PLE	product line engineering
RFLP	requirement, functional, logical, and physical
RQ	research question
SDV	software-defined vehicle
SoS	system of systems
SPLE	systems and software product line engineering
VIN	vehicle identification number

Appendix A. Mapping of Requirements to MSPLE Elements and Case Study Evidence

The following Table A1 provides the detailed evidence supporting the requirement evaluation presented in Section 8.1 by linking each requirement to the corresponding MSPLE elements and case study results.

Table A1. Mapping of requirements to MSPLE elements, case study evidence and results.

ID	MSPLE Element	Case Study Evidence	MSPLE
R-01	RFLP layers	Side mirror CAD, logical and functional models	●
R-02	RFLP architecture	Requirements, functions, logical and physical artifacts	●
R-03	Feature-based module description	Side mirror module, feature models in model versions	●
R-04	Module definition	No dedicated partitioning method defined	●
R-05	Modular systems architecture	Assembly of side mirror modules	●
R-06	Feature constraints, architecture relations	Dependency of modules and model versions	●
R-07	Feature catalog, attributes	Module-specific feature models/model versions	●
R-08	Not supported	Not demonstrated	○
R-09	Feature-based configuration	Side mirror variant derivation	●
R-10	Variant management	High/mid/low side mirror variants	●

●: Requirement satisfied; ●: Requirement partially satisfied; ○: Requirement not satisfied.

References

- Guissouma, H.; Klare, H.; Sax, E.; Burger, E. An Empirical Study on the Current and Future Challenges of Automotive Software Release and Configuration Management. In *Proceedings of the 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Prague, Czech Republic, 29–31 August 2018*; IEEE: New York, NY, USA, 2018; pp. 298–305.
- Zellmer, P.; Krüger, J.; Leich, T. Decision Making for Managing Automotive Platforms: An Interview Survey on the State-of-Practice. In *FSE 2024: Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering, Porto de Galinhas, Brazil, 15–19 July 2024*; d'Amorim, M., Ed.; ACM: New York, NY, USA, 2024; pp. 318–328.
- Guissouma, H.; Diewald, A.; Sax, E. *A Generic System for Automotive Software Over the Air (SOTA) Updates Allowing Efficient Variant and Release Management*; Springer: Cham, Switzerland, 2018.
- Frank, C.; Holsten, L.; Şahin, T.; Vietor, T. How to manage vehicle platform variants? A method to assess platform variance through competitive analysis. *Procedia CIRP* **2022**, *109*, 598–603. [\[CrossRef\]](#)
- Heeps, G. The Big Cell. *Electr. Hybrid Veh. Technol. Int.* **2020**, *2020*, 76–80. [\[CrossRef\]](#)
- Pohl, K.; Metzger, A. Variability management in software product line engineering. In *ICSE06: Proceedings of the 28th International Conference on Software Engineering, Shanghai, China, 20–28 May 2006*; Osterweil, L.J., Rombach, D., Soffa, M.L., Eds.; ACM: New York, NY, USA, 2006; pp. 1049–1050.
- ISO/IEC 26550; Software and Systems Engineering—Reference Model for Product Line Engineering and Management. International Organization for Standardization (ISO): Geneva, Switzerland, 2015. Available online: <https://www.iso.org/standard/69529.html> (accessed on 19 March 2026).
- Clements, P.; Northrop, L.M. *Software Product Lines: Practices and Patterns*; Addison-Wesley: Boston, MA, USA, 2001.
- Apel, S.; Batory, D.; Kästner, C.; Saake, G. *Feature-Oriented Software Product Lines*; Springer: Berlin/Heidelberg, Germany, 2013.
- Böckle, G.; Pohl, K.; van der Linden, F. A Framework for Software Product Line Engineering. In *Software Product Line Engineering*; Pohl, K., Böckle, G., van der Linden, F., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 19–38.
- Horcas, J.-M.; Pinto, M.; Fuentes, L. Software Product Line Engineering. In *SPLC 2019: Proceedings of the 23rd International Systems and Software Product Line Conference—Volume A, Paris France, 9–13 September 2019*; Berger, T., Collet, P., Duchien, L., Fogdal, T., Heymans, P., Kehrer, T., Martinez, J., Mazo, R., Montalvillo, L., Salinesi, C., et al., Eds.; ACM: New York, NY, USA, 2019; pp. 164–176.
- Lindebauer, Y.; von Eisebeck, R.; Vietor, T. Automotive software product lines for ECU software configuration: A systematic literature review. *J. Syst. Softw.* **2026**, *234*, 112716. [\[CrossRef\]](#)
- Thörn, C.; Sandkuhl, K. Feature Modeling: Managing Variability in Complex Systems. In *Complex Systems in Knowledge-Based Environments: Theory, Models and Applications*; Tolks, A., Jain, L.C., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 129–162.
- Clarke, D.; Helvensteijn, M.; Schaefer, I. Abstract delta modeling. *ACM SIGPLAN Not.* **2011**, *46*, 13–22. [\[CrossRef\]](#)
- Schäuffele, J.; Zurawka, T. *Automotive Software Engineering*; Springer: Wiesbaden, Germany, 2024.
- ISO/IEC 26580; Software and Systems Engineering—Methods and Tools for the Feature-Based Approach to Software and Systems Product Line Engineering. International Organization for Standardization (ISO): Geneva, Switzerland, 2021. Available online: <https://www.iso.org/standard/43139.html> (accessed on 19 March 2026).

17. Kang, K.; Cohen, S.; Hess, J.; Novak, W.; Peterson, A. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*; Defense Technical Information Center (DTIC): Fort Belvoir, VA, USA, 1990.
18. von Eisebeck, R.; Lindebauer, Y.; Vietor, T. Variantenkomplexität in der automobilen Steuergerätekonfiguration: Status quo und Potenziale durch SPLE. In *DS 140: Proceedings of the 36th Symposium Design for X (DFX2025), Hamburg, Germany, 11–12 September 2025*; The Design Society: Glasgow, UK, 2025; pp. 199–208.
19. Salvador, F. Toward a Product System Modularity Construct: Literature Review and Reconceptualization. *IEEE Trans. Eng. Manag.* **2007**, *54*, 219–240. [[CrossRef](#)]
20. Krause, D.; Gebhardt, N. Modulare Produktstrukturstrategien. In *Methodische Entwicklung modularer Produktfamilien*; Krause, D., Gebhardt, N., Eds.; Springer: Berlin/Heidelberg, Germany, 2018; pp. 127–155.
21. Beckmann, G.; Gebhardt, N.; Bahns, T.; Krause, D. Approach to Transfer Methods for Developing Modular Product Families Into Practice. In *DS 84: Proceedings of the DESIGN 2016, 14th International Design Conference, Dubrovnik, Croatia, 16–19 May 2016*; The Design Society: Glasgow, UK, 2016; pp. 1185–1194.
22. Blees, C.; Jonas, H.; Krause, D. Perspective-Based Development of Modular Product Architectures. In *DS 58-4: Proceedings of ICED 09, the 17th International Conference on Engineering Design, Palo Alto, CA, USA, 24–27 August 2009*; The Design Society: Glasgow, UK, 2009; Volume 4, pp. 95–106.
23. Kusiak, A.; Huang, C.-C. Development of modular products. *IEEE Trans. Compon. Packag. Manuf. Technol. Part A* **1996**, *19*, 523–538. [[CrossRef](#)]
24. Kaplan, R.S.; Anderson, S.R. Time-Driven Activity-Based Costing. *SSRN J.* **2003**. [[CrossRef](#)]
25. Jänsch, J.; Birkhofer, H. The Development of the Guideline VDI 2221—The Change of Direction. In *DS 36: Proceedings Design 2006, the 9th International Design Conference, Dubrovnik, Croatia, 15–18 May 2006*; The Design Society: Glasgow, UK, 2006; pp. 45–52.
26. Blees, C. *Eine Methode zur Entwicklung Modularer Produktfamilien*; TUHH Universitätsbibliothek: Hamburg, Germany, 2011.
27. Krause, D.; Ripperda, S. An assessment of methodical approaches to support the development of modular product families. In *DS 75-4: Proceedings of the 19th International Conference on Engineering Design (ICED13), Design for Harmonis*; The Design Society: Glasgow, UK, 2013; Volume 4, pp. 31–40.
28. Blees, C.; Kipp, T.; Beckmann, G.; Krause, D. Development of modular product families: Integration of design for variety and modularization. In *Proceedings of NordDesign 2010, the 8th International NordDesign Conference*; The Design Society: Glasgow, UK, 2010; pp. 159–170.
29. Martin, M.V.; Ishii, K. Design for variety: Developing standardized and modularized product platform architectures. *Res. Eng. Des.* **2002**, *13*, 213–235. [[CrossRef](#)]
30. Berschik, M.C.; Krause, D. Towards Model-Based Modular Product Families in the Context of Systems Engineering. In *Proceedings of NordDesign 2024, Reykjavik, Iceland, 12–14 August 2024*; The Design Society: Glasgow, UK, 2024; pp. 71–78.
31. Hann, M.; Woeller, L.-N.; Krause, D. Model-Based Consistency for Design for Variety and Modularization. In *Proceedings of the 20th International DSM Conference, Trieste, Italy, 15–17 October 2018*.
32. Ezzat, O.; Medini, K.; Boucher, X.; Delorme, X. Product and service modularization for variety management. *Procedia Manuf.* **2019**, *28*, 148–153. [[CrossRef](#)]
33. Alkahtani, M.; Helo, P.; Piya, S.; Shamsuzzoha, A. Modular product architecture to manage product development complexity. *Int. J. Ind. Syst. Eng.* **2020**, *36*, 225–247. [[CrossRef](#)]
34. Damiani, F.; Schaefer, I.; Winkelmann, T. Delta-oriented multi software product lines. In *SPLC '14: Proceedings of the 18th International Software Product Line Conference—Volume 1, Florence Italy, 15–19 September 2014*; Gnesi, S., Heymans, P., Rubin, J., Czarnecki, K., Dhungana, D., Fantechi, A., Eds.; ACM: New York, NY, USA, 2014; pp. 232–236.
35. Trujillo-Tzanahua, G.I.; Juárez-Martínez, U.; Aguilar-Lasserre, A.A.; Cortés-Verdín, M.K. Multiple Software Product Lines: Applications and challenges. In *Trends and Applications in Software Engineering*; Mejia, J., Muñoz, M., Rocha, Á., Quiñonez, Y., Calvo-Manzano, J., Eds.; Springer: Cham, Switzerland, 2018; pp. 117–126.
36. Mejia, J.; Muñoz, M.; Rocha, Á.; Quiñonez, Y.; Calvo-Manzano, J. (Eds.) *Trends and Applications in Software Engineering*; Springer: Cham, Switzerland, 2018.
37. Eklund, U.; Gustavsson, H. Architecting automotive product lines: Industrial practice. *Sci. Comput. Program.* **2013**, *78*, 2347–2359. [[CrossRef](#)]
38. Oliinyk, O.; Petersen, K.; Schoelzke, M.; Becker, M.; Schneickert, S. Structuring automotive product lines and feature models: An exploratory study at Opel. *Requir. Eng.* **2017**, *22*, 105–135. [[CrossRef](#)]
39. Pavalkis, S. Towards Industrial Integration of MBSE into PLM for Mission-Critical Systems. *INCOSE Int. Symp.* **2016**, *26*, 2462–2477. [[CrossRef](#)]
40. Madeira, R.H.; de Sousa Pinto, D.H.; Forlingieri, M. Variability on System Architecture using Airbus MBPLE for MOFLT Framework. *INCOSE Int. Symp.* **2023**, *33*, 601–615. [[CrossRef](#)]
41. Lamah, J.; Dubray, A.; Jankovic, M. Modeling variability in product line engineering (PLE) for systems engineering (SE). *Proc. Des. Soc.* **2025**, *5*, 2491–2500. [[CrossRef](#)]

42. de Sousa Pinto, D.H. Airbus: MBPLE in Commercial Aviation. In *Model-Based Product Line Engineering (MBPLE)*; Forlingieri, M., Weilkens, T., Chalé-Gongora, H.G., Eds.; Wiley: Hoboken, NJ, USA, 2025; pp. 233–244.
43. Holmqvist, T.; Persson, M. *Modularization—Not Only a Product Issue*; The Design Society: Glasgow, UK, 2004; pp. 25–36.
44. Shamsuzzoha, A.; Helo, P.T.; Kekale, T. Application of modularity in world automotive industries: A literature analysis. *Int. J. Automot. Technol. Manag.* **2010**, *10*, 361. [\[CrossRef\]](#)
45. Lampón, J.F.; Cabanelas, P.; Frigant, V. *The New Automobile Modular Platforms: From the Product Architecture to the Manufacturing Network Approach*; Munich Personal RePEc Archive (MPRA): Munich, Germany, 2017.
46. Lampón, J.F.; Frigant, V.; Cabanelas, P. Determinants in the adoption of new automobile modular platforms. *J. Manuf. Technol. Manag.* **2019**, *30*, 707–728. [\[CrossRef\]](#)
47. Albers, A.; Bursac, N.; Scherer, H.; Birk, C.; Powelske, J.; Muschik, S. Model-based systems engineering in modular design. *Des. Sci.* **2019**, *5*, e17. [\[CrossRef\]](#)
48. Thiel, S.; Babar, M.A.; Botterweck, G.; O'Brien, L. Software Product Lines in Automotive Systems Engineering. *SAE Int. J. Passeng. Cars-Electron. Electr. Syst.* **2009**, *1*, 531–543. [\[CrossRef\]](#)
49. Flores, R.; Krueger, C.; Clements, P. Mega-scale product line engineering at General Motors. In *SPLC '12: Proceedings of the 16th International Software Product Line Conference—Volume 1, Salvador Brazil, 2–7 September 2012*; Santana de Almeida, E., Schwanninger, C., Benavides, D., Eds.; ACM: New York, NY, USA, 2012; pp. 259–268.
50. Wozniak, L.; Clements, P. How automotive engineering is taking product line engineering to the extreme. In *SPLC '15: Proceedings of the 19th International Conference on Software Product Line, Nashville, TN, USA, 20–24 July 2015*; Schmidt, D.C., Ed.; ACM: New York, NY, USA, 2015; pp. 327–336.
51. Blessing, L.T.; Chakrabarti, A. *DRM, A Design Research Methodology*; Springer: London, UK, 2009.
52. Porsche, A.G. Porsche Strategie 2030 Plus. Available online: <https://investorrelations.porsche.com/de/ueber-uns/strategie> (accessed on 9 April 2026).
53. Holsten, L.; Frank, C.; Krüger, J.; Leich, T. Electrics/Electronics Platforms in the Automotive Industry: Challenges and Directions for Variant-Rich Systems Engineering. In *Proceedings of the 17th International Working Conference on Variability Modelling of Software-Intensive Systems. VaMoS 2023, Odense, Denmark, 25–27 January 2023*; Cohen, M., Thüm, T., Mauro, J., Eds.; ACM: New York, NY, USA, 2023; pp. 50–59.
54. ISO 26262-1:2018-12; Road Vehicles—Functional Safety—Part 1: Vocabulary. International Organization for Standardization (ISO): Geneva, Switzerland, 2018. Available online: <https://www.dinmedia.de/de/norm/iso-26262-1/300423910> (accessed on 3 July 2026).
55. Bursac, N.; Albers, A.; Schmitt, T. Model Based Systems Engineering in Modular Design—A Potential Analysis using Portal Type Scraper Reclaimers as an Example. *Procedia CIRP* **2016**, *50*, 802–807. [\[CrossRef\]](#)
56. Visser, R.; Basson, A.H.; Kruger, K. An architecture for the integration of product and production digital twins: An automotive case study. *Softw. Syst. Model.* **2026**, *25*, 757–785. [\[CrossRef\]](#)
57. Bender, B.; Gericke, K. *Pahl/Beitz Konstruktionslehre*; Springer: Berlin/Heidelberg, Germany, 2021.
58. ISO/IEC 25002:2024; Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE): Quality Model Overview and Usage. International Organization for Standardization (ISO): Geneva, Switzerland, 2024. Available online: <https://www.iso.org/standard/78175.html> (accessed on 20 May 2026).
59. Krog, J.; Şahin, T.; Vietor, T. Towards a Systems Engineering Methodology for Architecture Development of Vehicle Concepts. In *How Product and Manufacturing Design Enable Sustainable Companies and Societies, Proceedings of NordDesign 2022, Copenhagen, Denmark, 16–18 August 2022*; The Design Society: Glasgow, UK, 2022; p. 12.
60. Krog, J.; Akbas, C.; Nolte, B.; Vietor, T. Development of a Functional and Logical Reference System Architecture in Automotive Engineering. *Systems* **2025**, *13*, 141. [\[CrossRef\]](#)
61. Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M.; Regnell, B.; Wesslén, A. *Experimentation in Software Engineering*; Springer: Berlin/Heidelberg, Germany, 2024.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.