The Terminology of Automotive Product-Structuring Concepts: A Systematic Mapping Study

Philipp Zellmer, Lennart Holsten, Jacob Krüger, and Thomas Leich

Abstract—**The automotive industry is undergoing a significant transformation, with vehicles evolving into complex, interconnected cyber-physical systems. This transformation is caused by new customer demands, legal standards, and technological innovations, which lead to an increasing amount of electronic control units, software, and features. To address the consequent software-related challenges, automotive manufacturers are adopting methodologies like software product-line engineering, electrics/electronics platforms, and product generation engineering. However, each of these methodologies relies on an own vocabulary, necessitating a unification of the divergent understandings and interpretations of key terms and definitions. In this article, we investigate and discuss a terminological framework that provides a common ground for specifying a unified product-structuring concept. For this purpose, we conducted a systematic mapping study to develop a framework of existing terms and definitions used to describe product-structuring concepts in software, electrics/electronics, as well as mechanical engineering. We discuss the differences and commonalities of the terminologies to help practitioners in integrating and applying product-structuring concepts as well as to guide future research.**

Index Terms—**automotive, electrics/electronics, product line, life-cycle management, cyber-physical system, product-structuring concept**

I. Introduction

To remain competitive, automotive manufacturers must continuously evolve their product portfolios by integrating new features into their vehicles. In the past, vehicles and innovative features were centered around hardware components. However, technological advances, changing customer demands, and legal standards necessitate the integration of a rising number of software features into the existing hardware platforms. In fact, most innovative features in modern vehicles are rooted in software rather than hardware, due to prevalent trends like autonomous driving, driver assistance systems, electrification, and vehicle connectivity [6], [11], [81]. Logically, vehicles are also becoming more and more digitized, and have essentially transitioned towards software-intensive cyber-physical systems that require effective interactions between hardware and software to deliver innovative features [37], [70].

The increasing number of software features poses challenges for automotive manufacturers, particularly when it comes to engineering and managing their vehicle platforms. In the past, manufacturers developed hardware platforms that were build around mechanical components, thereby engineering a unified architecture for diverse vehicles to benefit from reuse and standardization [34]. Moving towards more software-intensive platforms and engineering processes was necessary, but also challenging—particularly when the old hardware platforms should still be reused while integrating heavily software-focused features like over-the-air (OTA) updates or self-driving capabilities.

Due to the growing complexity of vehicle platforms, it becomes progressively more challenging to effectively manage the variability of all hardware and software artifacts along with their intricate interconnections. For this reason, automotive manufacturers are facing disproportionately increasing expenses and efforts. To tackle such problems, the manufacturers adopt product-structuring concepts and methods that consider vehicles as software-intensive cyber-physical systems. In particular, automotive manufacturers have started to use variant-management concepts stemming from software product-line engineering [17], [65], [85] to incorporate the software perspective into their established hardware-platform strategies [23], [106].

However, adding a software perspective on top of a static hardware platform does not solve the actual problems the digital transformation of vehicles poses. Instead, a holistic platform strategy that considers all dimensions of modern vehicles (hardware, software, electrics/electronics) as well as their interconnections is needed. Developing such a holistic platform strategy is a complex and challenging task that, first of all, requires a unification of the divergent understandings and interpretations of key terms and definitions across the involved domains. In this article, we aim to tackle this first step by triangulating and discussing such a terminology to contribute a common ground for developing and implementing unified product-structuring concepts. To derive this terminology, we have built on our previous systematic mapping study of product-structuring concepts [119], which we have expanded and shifted towards this novel research goal. Based on our new study, we make the following, completely novel, contributions in this article:

- We report a mapping study covering 40 publications through which we collected the varying terms and definitions used within product-structuring concepts that are established in three different domains (Section IV).
- We synthesize and discuss the key terms we extracted

P. Zellmer and L. Holsten are with Volkswagen AG, Wolfsburg & Harz University, Wernigerode, Germany (philipp.zellmer2@volkswagen.de | lennart.holsten@volkswagen.de).

J. Krüger is with Eindhoven University of Technology, Eindhoven, The Netherlands (j.kruger@tue.nl).

T. Leich is with Harz University, Wernigerode, Germany (tleich@hs-harz.de).

from the publications (Section V) and illustrate their relations in a conceptual model (Section VI).
- We propose a framework that consolidates the key definitions and outlines their relationships for researchers and practitioners to build upon (Section VI).

Our results can help practitioners identify product-structuring concepts for their product portfolios, provide a common terminology to avoid confusions, and display the connections between different terms. For researchers, our work provides a basis for developing new integrated product-structuring concepts, which further guide the design of new support techniques based on a unified understanding. Overall, we hope that our contributions lay the foundation for new software and systems engineering research that can be directly applied by and benefit practitioners.

II. Background and Related Work

Next, we introduce the background of our work. Specifically, we discuss current trends in the automotive industry (Section II-A) and product-structuring concepts (Section II-B).

A. Automotive Systems Engineering

Automotive Innovation. The automotive industry faces increasing demands regarding functional safety and security, onboard communication, comfort, as well as environmental sustainability; leading to a continuously growing amount of vehicle features [1], [15], [114]. In parallel, software has evolved into the primary driver for innovative features within the automotive domain, solidifying its role as a key factor for competitive advantages [6], [11]. As a result, there has been a swift and almost exponential annual increase of software integrated into a vehicle [19], [81]. By now, a substantial share (approximately 80 % to 90 %) of innovations within the automotive industry stems from electronic advancements that predominantly rely on software [81]. This trend is further reflected by a notable increase in the lines of code in a vehicle, particularly premium ones, which exceed 100 million lines of codes [19]. With the considerable potential of software in facilitating innovation and cost-effective prototyping, vehicles will continue to become more complex as the number of features as well as associated electronic control units (ECUs), sensors, and actuators keeps growing [5], [6], [9], [14], [114]. Consequently, automotive manufacturers are facing increasing development costs as well as new variability-induced challenges in managing their extensive product portfolios—which include reusable software, ECUs, and hardware that yield numerous customizeable vehicles across various vehicle generations [18].

Platform Engineering. Pursuing effective product-portfolio development and management, automotive manufacturers have implemented platform strategies as an instrument for variability management [8], [99]. The core idea of automotive platforms is built around a simple premise: Rather than developing and evolving each vehicle model individually, essential vehicle components are consolidated into a (hardware) platform. Such a platform is developed once and then deployed across multiple vehicle models, fostering reuse and enhancing overarching synergies [21], [45], [75], [92], [101], [112]. Despite the advancing digitization of vehicles, mechanical components continue to dominate automotive platforms, which is why automotive manufacturers have recently increased their efforts to integrate concepts from software engineering into their established frameworks [35]. In fact, software product lines utilize a comparable strategy and have been inspired by automotive hardware platforms. A software product line integrates reusable software artifacts as well as their variation points into a unified platform to streamline software management [17], [65], [85]. Systematically utilizing the resulting software platform can significantly contribute to reducing time-to-market, decreasing costs, and improving software quality by facilitating the reuse as well as standardization of software artifacts [62], [63], [98], [110]. Despite such recognized advantages of a software platform, automotive manufacturers struggle to implement such a platform consistently across their entire product portfolio, primarily due to the persisting reliance on hardware platforms. Moreover, the intricate interconnections and distributions of different hardware, software, and electrics/electronics components across various manufacturers as well as suppliers emphasize the demand for integrated engineering methodologies that acknowledge vehicles as cyber-physical systems [48].

Cyber-Physical Systems. The concept of cyber-physical systems outlines complex systems with closely interconnected physical and software components that interact based on their operational context and environment. Specifically, the primary objective is to monitor and control physical devices within the system through digital communication [60], [68], [109], [115]. Across an expanding range of industries, cyber-physical systems are emerging as a catalyst for innovation, showcasing the potential to evolve beyond today's information systems [68], [84], [95]. Cyber-physical systems are already employed across diverse domains, including high-confidence medical devices, production systems, and critical infrastructure control [68], [89].

The automotive industry directs notable resources towards advancing the intelligence of both, its vehicles and its production systems—emphasizing the increased integration of connectivity, electronics, and software components [60], [87]. Consequently, modern vehicles are evolving into complex distributed cyber-physical systems that are characterized by more than a hundred heterogeneous processors, interconnected subsystems with diverse sensors and actuators, multiple radio interfaces, as well as connections to other vehicles, the infrastructure, or back-end systems [60], [87]. This allows to integrate highly innovative features, including intelligent mobility assistants, smart home applications, and x-by-wire systems that can utilize data from the vehicles, their underlying infrastructure, or back-end systems [51], [67], [87].

Automotive Life Cycle Management. Within the framework of product life cycle management, a comprehensive and centralized system is established to govern a product from its initiation to its disposal or retirement [40], [42], [105]. Depending on the industry, the duration and content of product life cycles can vary greatly, with factors like product innovation and consumer behavior playing pivotal roles [90], [121].

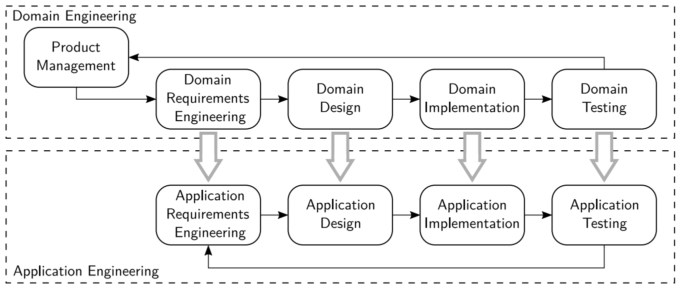

Fig. 1: Software product-line engineering based on Pohl et al. [85] and adapted from Krüger [62].

Strategic efforts across industries to achieve multi-market saturation and competitive edges involve expanding product portfolios through the introduction of numerous derivatives and variants. This trend leads to an increased incorporation of electrical, electronic, and software components to facilitate effective communication between systems [54]. The resulting cyber-physical systems are characterized by increased complexity, requiring extended management throughout their life cycles [22], [54].

In response to the growing reliance on software in the automotive industry, the capability to update vehicles with novel features or to fix identified issues becomes more and more important to fulfill customer needs and requirements [49], [71], [83], [121]. The high number of devices underscores the limitations of traditional update processes through service centers, as they tend to be time-consuming, inefficient, and troublesome [20], [38], [54]. Addressing this challenge, the automotive industry is heavily investing in OTA software updates to enable remote modifications of vehicle features or bug fixes, thereby enhancing the efficiency and scalability of these updates [20], [38], [54]. Today, OTA updates are instrumental for the automotive industry for achieving customer benefits through software updates throughout a vehicle's life cycle. In this article, we refer to this new life cycle management within the automotive industry as "software life cycle management," which describes life cycle management via OTA software updates [118].

B. Product-Structuring Concepts

For the purpose of this study, we define a product-structuring concept as a methodology designed to systematically manage an extensive product portfolio consisting of related, yet customized products. In the following, we outline common product-structuring concepts relevant to our study, acknowledging that numerous domain-specific strategies (e.g., clone-and-own management for software variants [62], [65], [93], [104]) besides these exist. These concepts are of particular importance to our work, as they offer prospects for integration into a complex platform involving hardware, software, and ECUs.

Software Product-Line Engineering. Product-line engineering revolves around the premise that a group of similar products shares a defined set of core assets that can be explicitly specified and reused among these products, forming a customizable platform. Enhanced reuse and standardization within the product line foster synergies among individual product variants. In the context of software engineering, software product lines have emerged as a pivotal concept for managing variability in software-intensive systems [17], [50], [69], [85]. As we illustrate in Figure 1, software product-line engineering encompasses two processes: domain engineering and application engineering [30], [52], [65], [85]. Domain engineering involves developing core assets, entailing all software artifacts and their interconnections being consolidated into a cohesive software platform. Within a software platform, the reusable artifacts are accompanied by specifications outlining the constraints between these artifacts to ensure that a concrete product variant can be derived [69], [74], [88]. Application engineering is the process of configuring and deriving concrete products from the software platform to meet distinct customer requirements [69], [106]. Through a systematic variability-management framework that emphasizes increased reuse and standardization, software product lines can substantially reduce costs, enhance software quality, and expedite time-to-market [23], [26], [62], [63], [98], [103], [110].

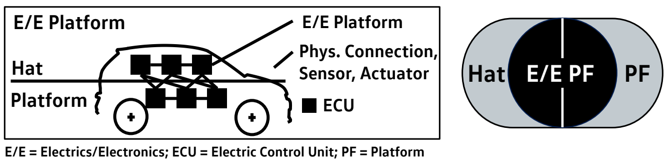
E/E Platform
Hat
Platform
E/E Platform
Phys. Connection, Sensor, Actuator
■ ECU
Hat
E/E PF
PF
E/E = Electrics/Electronics; ECU = Electric Control Unit; PF = Platform

Fig. 2: The electrics/electronics platform concept based on Holsten et al. [43].

Electrics/Electronics Platform Engineering. Following the concept of engineering an integrated platform, electrics/electronics platforms have been proposed as a method to integrate a common set of vehicle components into a unified architecture that is applicable across multiple vehicle models. Diverging from hardware or software platforms, electrics/electronics platform engineering emphasizes the integration of software and hardware artifacts within a comprehensive electrics/electronics architecture. To illustrate how this concept can be applied in the automotive sector, we present the electrics/electronics platform alongside its interconnections with the established hardware platform and hat strategy in Figure 2 [43], [48], [86]. Rather than delineating between mechanical components (hardware platform) and customer-relevant components (hat), the electrics/electronics platform concept integrates all electrics/electronics components of the relevant vehicles into a unified layer. This involves both the entirety of software artifacts and their physical implementation through embedded ECUs. Functioning as a comprehensive interface layer, the electrics/electronics platform establishes a core electrics/electronics architecture, closely connecting software and hardware artifacts; recognizing the vehicle as an integrated cyber-physical system. Consequently, inherent advantages linked to platform engineering, including increased reuse and overall synergies, can be maximized across the entirety of software-related vehicle components. Successfully implementing an electrics/electronics platform builds on several factors, such

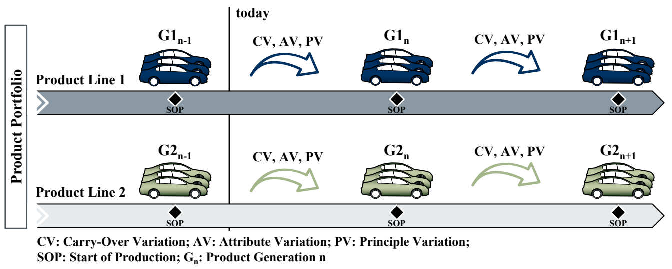
today
Product Portfolio
Product Line 1
Product Line 2
G1$_{n-1}$
G1$_n$
G1$_{n+1}$
G2$_{n-1}$
G2$_n$
G2$_{n+1}$
CV, AV, PV
SOP
CV: Carry-Over Variation; AV: Attribute Variation; PV: Principle Variation;
SOP: Start of Production; G$_n$: Product Generation n

Fig. 3: The product-generation engineering concept based on Alberts et al. [3].

as efficient reuse and standardization as well as the platform's adaptability (e.g., to varying sales markets, equipment lines, or technological developments) [48], [86].

Product-Generation Engineering. Product-generation engineering builds on the premise that mechatronic products, particularly in the context of modern vehicles, are rarely initiated from scratch. Instead, they typically evolve from an existing product known as the reference product [2], [3]. An analogous strategy is widely recognized in software engineering, commonly referred to as clone-and-own development [62], [65], [93], [104]. The engineering process for a new product generation involves incorporating existing components and systems from the reference product, while also developing and integrating new subsystems. Distinct types of variations can be defined: carry-over variation, attribute variation, and principle variation (cf. Figure 3). Carry-over variation involves adopting individual elements from the reference product, with adjustments to fulfill interface specifications. For instance, existing technical solutions of the reference product may be adapted and integrated into a new vehicle generation. Attribute variation encompasses changing specific vehicle attributes, such as adaptations to the geometrical shapes of component or to functional parameters, while maintaining the underlying technical and functional concept of the reference product. Principal variations involve engineering activities that add or remove elements or links within product-generation engineering. This includes new or adjusted vehicle features, manufacturing process adaptations, and additional software artifacts with their links [4]. Considering the various types of variations, improvements in reuse and standardization across consecutive vehicle generations can be achieved. In addition, comprehensive synergies across various vehicle models can be leveraged by using the reference product as a basis for multiple products or entire product lines. In recent research, product-generation engineering has been refined to integrate the growing significance of software and digitization in vehicles. For this purpose, product-generation engineering is applied to vehicle functions by establishing functional roadmaps, which aim to map functional evolution across the entire product portfolio and life cycle [3], [24].

Related Work. We are aware of four publications that overlap with our systematic mapping study on automotive product-structuring concepts. Nevertheless, none of these publications presents a systematic mapping that fulfills our research objectives. Specifically, Marchezan et al. [72] conducted a systematic literature review on scoping techniques for product-line engineering. Kenner et al. [53] report a systematic mapping study covering safety and security techniques in the context of configurable software systems. Galster et al. [32] describe a systematic literature review that focuses on variability handling in software engineering. While these studies do not specifically cover the automotive domain or other product-structuring concepts, they collected relevant publications regarding managing software product-line engineering. In another work, Knieke et al. [59] report a systematic literature review on holistic approaches to address the evolution of automotive software product-line architectures. Analyzing a total of 107 papers, Knieke et al. discuss automotive software product lines, but do not address papers related to other product-structuring concepts. While their investigation aligns with our research, our emphasis extends beyond software product lines. Within our systematic mapping study, we aim to provide an overview of existing terms and definitions used across product-structuring concepts, analyzing and discussing differences and commonalities to guide further research about a comprehensive terminological framework.

III. Initial Systematic Mapping Study

In 2023, we [118] conducted an initial systematic mapping study to gather an overview of existing product-structuring concepts and methods that consider both hardware and software artifacts and are applicable within the automotive domain. Our main objective was to provide an overview and enhance the understanding of existing research, focusing on different product-structuring concepts and their practical application in the automotive domain. More precisely, we:

- Identified, reviewed, and compared 17 publications to provide an overview of recent automotive product-structuring concepts.
- Conducted an in-depth analysis and discussion of key issues and lessons learned regarding the practical usability of these concepts.
- Defined potential areas for further research to guide the development of a feasible product-structuring concept that encompasses software, hardware, and their interdependencies.

As part of our analysis, we identified three product-structuring concepts that fulfill current automotive requirements: software product lines, electrics/electronics platforms, and product-generation engineering. We considered each as promising to align with current automotive industry trends. However, for each concept, we also observed several challenges and issues regarding their practical implementation. First, software product-line engineering provides possibilities for software-related automotive variability management, but its practical application is currently somewhat limited to subsystem level—due to lacks of real-world documentation of variability and of tool support. Second, electrics/electronics platforms enable the required combination of software, hardware, and mechanics within an integrated concept. However, their practical application has not yet been evaluated. Finally, product-generation engineering has proven to efficiently support automotive life-cycle management, but the concept lacks focus on software and research is at an early state.

Moreover, a lack of tool support as well as insufficient knowledge management have been reported as overarching challenges that hinder the practical implementation of all three concepts. As a result, decision-makers often miss specific guidance to find feasible solutions to practical problems. This is also related to the concepts being mostly presented at a rather generic level, involving broad recommendations when applying them. In our initial mapping study, we derived several lessons from such reported limitations and challenges: While software product-line papers emphasize the improvement of variability management techniques through concrete methodologies and consistent processes, product-generation engineering papers focus on transforming automotive development processes towards systematic functional orientation. Consequently, we deduced that the reported experiences were rarely translated into concrete practical guidance, leading to limited decision support activities.

Comparing the investigated product-structuring concepts, we reasoned that software product-line engineering, hardware platforms, and electrics/electronics platforms follow the same basic idea: integrating common assets into an overarching platform. Referring to this platform, all three concepts aim to attain synergistic effects through improved reuse and standardization, facilitating the derivation of individual products tailored to specific customer requirements (cf. Figure 4). We argued that the same terms are defined differently across diverse domains and differing terminologies exist to describe the same concepts. Consequently, we proposed to work towards an overarching concept that integrates established software-centric platform concepts with functional roadmaps as used in product-generation engineering. This overarching concept aims to address current and future requirements within the automotive industry, promoting particularly the practical applicability of the concept. To summarize, we presented the following key findings:

- Software product-line engineering, hardware platforms, and electrics/electronics platforms follow the same basic platform idea.
- The practical application and a lack of tool support were reported as key challenges across all three of these product-structuring concepts.
- The lack of practical applicability demands for further research towards an integrated concept that combines software orientation and practical feasibility.

Based on these results, we intended to identify existing definitions and terminologies regarding key terms of automotive product-structuring concepts and to compare them across the domains of software engineering, electrics/electronics engineering, and mechanical engineering.

To further underpin the necessity for a unified framework, we provide an example from automotive practice based on the first two authors' work experience. As outlined before, Volkswagen AG has increasingly adapted concepts based in software engineering in recent years, such as introducing electrics/electronics platforms. Today, variant management

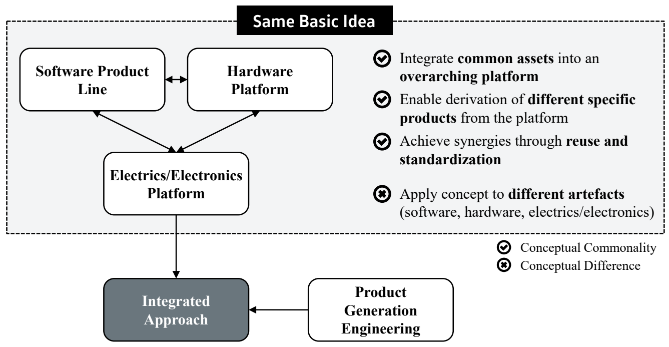
Same Basic Idea
Software Product Line
Hardware Platform
Electrics/Electronics Platform
Integrate **common assets** into an **overarching platform**
Enable derivation of **different specific products** from the platform
Achieve synergies through **reuse and standardization**
Apply concept to **different artefacts** (software, hardware, electrics/electronics)
Conceptual Commonality
Conceptual Difference
Integrated Approach
Product Generation Engineering

Fig. 4: The overview of conceptional connections between the concepts we identified in our initial mapping study [118].

within the company has to include activities dedicated to minimizing electrics/electronics platform variance. However, in applying such activities, we have observed that, for instance, the term "electrics/electronics platform variant" is defined and understood differently across various departments and areas. Establishing a common understanding, which is essential for a consistent complexity-reduction strategy, has emerged as a significant challenge for the company. Without such an understanding, decision-making processes may get delayed and transparency concerning the impact of variant decisions is limited. Consequently, variant management may suffer from operational inefficiencies, causing maintenance efforts to increase during the vehicle life-cycle [44], [120]. This real-world case exemplifies that the lack of a unified terminological framework can pose risks concerning consistent decision-making processes, time-to-market, and managing complexity.

IV. Methodology

Building on our initial mapping study on product-structuring concepts for automotive platforms [118], we aimed to investigate commonalities and differences of key terms and definitions stemming from domain-specific perspectives. For this purpose, we conducted a systematic mapping study following the guidelines for software-engineering research proposed by Kitchenham and her colleagues [56]–[58]. Next, we explain our research procedure, which we illustrate in Figure 5.

A. Research Questions

The increasing digitization and networking of vehicles has given rise to novel challenges, which are driven by increasing complexity, demands for technical innovations, software-based vehicle maintenance, and increasing concerns regarding cyber security. As a result, automotive companies are rapidly adopting methods from software engineering, carefully considering the specific challenges and requirements of the automotive industry. In this article, we aim to asses commonalities and differences of key terms that are required for product-structuring concepts, namely software product-line engineering, electrics/electronics platform engineering, and product-generation engineering. To achieve this objective, we formulated the following research questions (RQs):

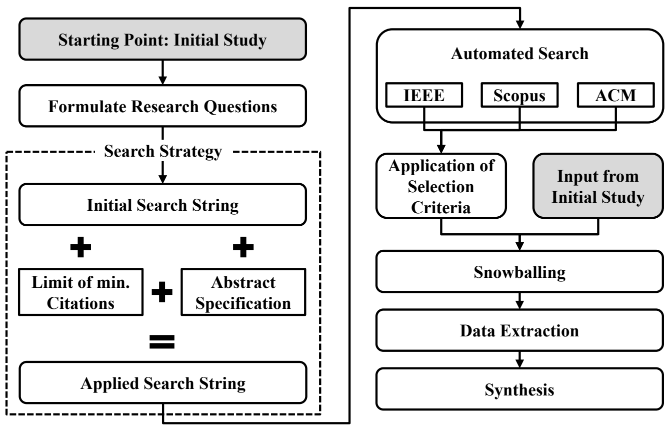

Fig. 5: Overview of our research methodology.

RQ$_1$ *What are the key terminologies required for automotive product-structuring concepts?*
We aimed to achieve a cross-domain overview of existing definitions regarding the most important concepts within automotive product-structuring concepts. Such an overview helps practitioners in seamlessly transferring theoretical concepts into practice. To answer this question, we extracted key definitions from high-impact literature to capture the current state of research regarding existing definitions in different domains.

RQ$_2$ *What terminological differences and commonalities exist between domains?*
Our objective was to assess the hypothesis that identical terms are defined differently across individual domains and that different terms exist to represent the same concepts. To tackle this question, we map the relations between the terminologies we found and structure them within a conceptual model.

RQ$_3$ *What practical challenges and implications can be derived from the results?*
Based on the cross-domain terminological comparison and our mapping, we discuss potential issues that automotive practitioners can encounter when implementing product-structuring concepts in practice. We further propose a comprehensive terminological framework and discuss how this could assist in mitigating these challenges.

Tackling these research questions contributes an overview of terms and definitions in the area of automotive product-structuring concepts for practitioners. This overview can further guide researchers in scoping new directions to solve real-world problems.

B. Search Strategy

Our search strategy built upon the results from our initial systematic mapping study and follows the same process. According to Kitchenham and Charters [58], selecting search terms and defining search resources is key for a systematic and reliable search. Based on our research questions and using the three key questions: “What?”, “How?”, and “What for?”, we defined key terms forming the basis of our search string.

TABLE I: Overview of synonyms and related terms.

“What?” Domain	“How?” Terms	“What for?” Concepts
automobile; car; vehicle; system; cyber physical; engineering; software	complex; variants; variety; variability; variable; life cycle; update; feature; release; function; baseline	e/e architecture; product line; product generation; e/e platform; electrics/electronics

Subsequently, we supplemented our terms with synonyms and related words, which we present in Table I. Each of the three key questions represents an individual string, combining the terms in Table I with OR operators. Furthermore, we added wildcards to account for stemming (“*”) and for small changes in writing (“?”). By connecting the resulting strings with an AND operator, we built the following search string:

(("automo*" OR "car" OR "vehicle*" OR "cyber*physical" OR "engineering" OR "software" OR "system") AND ("complex*" OR "varia*" OR "varie*" OR "life cycle" OR "update" OR "release" OR "feature" OR "baseline" OR "function") AND ("e/e?archite*" OR "product line" OR "product generation" OR "e/e?platform" OR "electri*/electronic*"))

We applied this search string to the full–text of papers listed in relevant databases to conduct test runs. Assuming that highly cited papers containing relevant terms and definitions could extend far into the past, we decided not to limit our search to a specific period. Not surprisingly, we found that the search string returned too many results, which is why we applied further criteria to reduce the search results. To assess the most relevant papers within the scientific community, we defined a limit of at least 150 citations (minimum) for a paper to consider it in our analysis, which we included in our selection criteria.

We deployed our final search string to the following digital libraries from computer science and engineering:

IEEE Xplore, which covers papers from electrical engineering, computer science, and electronics published by IEEE.

Scopus, which is a large database of peer-reviewed papers from various publishers with venues being included based on a quality assessment.

ACM Digital Library, which is a collection of full-text articles published by ACM and bibliographic records by other publishers covering computer science and information technology.

To apply our search strategy to the above databases, we modified our search string in terms of special characters and research area (e.g., limiting it to computer science and engineering on Scopus). Afterwards, we extended and improved the completeness of our search by inspecting our initial mapping study [118] again for relevant paper and performing backwards snowballing [64], [117]. Specifically, we analyzed the references listed in each selected primary study to identify further relevant papers that we may have missed during the automatic search.

C. Selection Criteria

To identify relevant papers to address our research questions, we defined the following inclusion criteria (ICs) for a

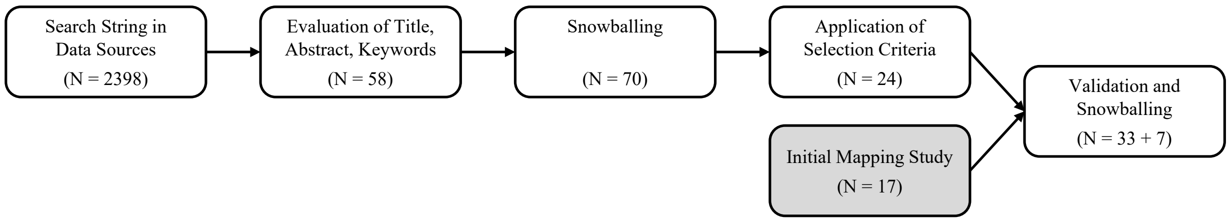
Search String in Data Sources (N = 2398)
Evaluation of Title, Abstract, Keywords (N = 58)
Snowballing (N = 70)
Application of Selection Criteria (N = 24)
Initial Mapping Study (N = 17)
Validation and Snowballing (N = 33 + 7)

Fig. 6: Stages of our search process and the number of papers we selected.

paper to fulfill:

- IC$_1$ The paper is concerned with product-structuring concepts in the automotive industry.
- IC$_2$ The paper has a minimum of 150 citations as a sign of being established within its domain.
- IC$_3$ The paper has been published in a peer-reviewed journal, conference, or workshop.
- IC$_4$ The paper has more than three pages to ensure it is an actual contribution and not just an abstract or summary.

Moreover, we defined the following exclusion criteria (ECs):

- EC$_1$ The paper is published in another language than English or German (we considered German, due to our proficiency and awareness of highly relevant work being published in it).
- EC$_2$ The paper is published only as a bachelor's thesis, master's thesis, or technical report.
- EC$_3$ The paper is published with incomplete or missing information about the publisher or publication type (i.e., we excluded gray literature).

Applying these selection criteria, we aimed to ensure that the selected papers fitted our research questions and were of appropriate quality.

D. Data Extraction

We extracted standard bibliographic data for each paper we identified, specifically, the source, author(s), title, publisher, publication year, and number of pages. To address our research questions, we extracted the following additional information:

- The domain from which it stems.
- A summary of the findings presented.
- A list of the research objectives.
- All terms and their definitions used.

We carefully studied the full text of each selected paper to extract this information. In Section V and Section VI, we report and discuss our results, respectively.

E. Conduct

In Figure 6, we display the conduct of our literature search, including the number of papers we ended up with after each step. We conducted the automated search between November 21, 2023, and November 29, 2023, recovering the number of results for each data source as we list in Table II. Overall, we retrieved 2,398 papers and subsequently screened titles, abstracts, and keywords of all papers to identify those relevant to our research questions based on our selection criteria.

TABLE II: Overview of the studies we identified from each data source.

Data Source	**Results**	**Selected**			
		SW	E/E	M	OD
IEEE Xplore	1,329	4	4	1	1
ACM Digital Library	457	3	1	1	0
Scopus	612	3	0	2	1
Snowballing	-	6	3	6	4
Sum	2,398	16	8	10	6

SW: Software Engineering;
E/E: Electrics/Electronics Engineering;
ME: Mechanical Engineering
OD: Other Domain

To improve the validity of our search, two authors cross-checked all publications. They resolved any disagreements by discussing the papers with respect to our research questions.

After this step, we kept 58 papers and performed backwards snowballing. Please note that we included books within our snowballing process in case these were referenced, and thus may contain important primary data. Then, by analyzing each paper in detail, we ended up with a set of 24 papers. From our initial mapping study [118], which covered 17 papers, we added seven that were also relevant to our study and fulfilled the selection criteria. The last step included another cross-validation, data extraction, and iterative snowballing on all papers we included so far. Finally, we obtained 40 relevant papers. We provide an overview of their sources in Table II.

F. Analysis

To analyze the extracted data, we built on our knowledge of the related work, the studies we identified, and our experiences from practice [43], [118], [119]. The practical experiences stem from the automotive industry in which the first two authors are working for several years. Both have been involved in various projects in one of the largest international automotive companies, Volkswagen AG. The first author is member of a project-management team, focusing on variant management, platform engineering, and software-portfolio management. The second author is situated in the life-cycle management department, which emphasizes software-change management and digital life-cycle management. As a consequence, they collaborate with various experts in these domains and discuss innovative concepts with these. Overall, the first two authors possess an in-depth understanding of contemporary concepts related to platform engineering and (digital) life-cycle management in practice. For the analysis of the extracted data, the authors applied their expertise to structure and classify it, using, for instance, card-sorting-like methods to unify terms and definitions. Additionally, all authors engaged in subsequent discussions to analyze and assess the data as well as findings.

V. Results (RQ_1 & RQ_2)

In this section, we present the main results of our mapping study, addressing RQ_1 and RQ_2. We selected a total of 40 papers, which we analyzed according to the criteria we defined for the data extraction. Specifically, we extracted all terms and definitions, using three levels to classify: *Primary definition available* (●), *Secondary definition available* (◐), and *No definition available* (○).

We only refer to definitions from secondary sources in cases in which we had no access to the actual primary sources. To verify that the terms and definitions are still in use, we performed a cross-check with recent papers. For this purpose, we identified contemporary papers that relate to each term using the extracted definitions [13], [24], [31], [36], [39], [59], [72], [80], [91]. Although most of the papers included in our study are not the most recent ones due to the inclusion criterion of a minimum of 150 citations, we could still ensure that the terms are still in use. We provide an overview of our resulting mapping in Table III.

Architecture. The term architecture is used across all domains and consistently describes a set of elements that fulfill specific requirements or implement certain functionalities [10], [25], [55], [82]. However, more specific definitions of an architecture are typically used in each domain. Within mechanical engineering, the term product architecture is commonly used, with Ulrich [108] establishing a widely adopted definition, highlighting three key properties: “(1) the arrangement of functional elements; (2) the mapping from functional elements to physical components; (3) the specification of the interfaces among interacting physical components.” The modular product architecture concept advances on this product architecture, aiming to enhance flexibility and reusability by emphasizing modularized components with standardized interfaces to address increasing product variety [10], [47], [76], [94], [108]. Consequently, we found a strong focus on physical components, their interconnections, and the resulting implementation of product functions within architecture definitions relating to mechanical engineering.

Within software engineering, we found several paper referring to “software architecture” or the “architecture of a software system.” Such a software architecture contains—analogous to product architectures—the software artifacts, components, and their interactions; displaying the overall topology and networking within the software system [12], [79], [82], [100]. So, the architecture addresses properties like compatibility and capacity at system level [100]. In addition, Broy et al. [12] utilize the term “hardware architecture” for automotive systems. Such a hardware architecture contains all physical devices that are involved in deploying the software onto a vehicle’s hardware components, for instance, sensors, actuators, and bus systems.

In summary, the concept architecture is mostly defined consistently in both mechanical engineering and software engineering, each having a domain-specific focus in terms of definitions. However, it is noteworthy that the respective terminology is strictly utilized within the specific domains. For instance, we identified no connections to software components within the context of product architectures.

Platform. The term platform is widely used and defined across engineering domains, with mechanical engineering and software engineering referring to different meanings. We found several mechanical-engineering papers defining the terms “platform” and “product platform” [28], [33], [47], [75], [77], [92], [108]. The mechanical platform follows a core idea: instead of developing each product individually, manufacturers define a set of components that are used across a family of products. Then, the resulting sub-assembly is referred to as “product platform.” However, besides this general definition, we found further specifications of a “product platform,” each emphasizing different priorities: Messac et al. [73] focus on modular platform structures to facilitate product customization and enable overarching production processes. Simpson et al. [102] extend the perspective on platforms by considering not only components, but also product parameters and features as part of a product platform. Muffatto [77] takes this a step further by introducing an organizational dimension next to the technical perspective and proposing that the “platform” can encompass cross-functional teams for implementing overarching technical platforms. The goal of introducing this definition is to optimize the existing overlap between technical and organizational platform aspects. Robertson and Ulrich [92] further extend the boundaries of the platform terminology beyond the technical perspective by including processes, knowledge, and relationships as integral shared assets of product platforms. Finally, Koufteros et al. [61] do not refer to product families, but designate a “core product” as the platform. This product serves as the technical basis for multiple product generations, resulting in synergies and reuse over the products’ evolution.

Within software engineering and electrics/electronics engineering, we observed a distinction between software and hardware platforms. The terminology of software platforms aligns with that of product platforms, both being defined as “a collection of reuseable artifacts” [85] fulfilling specific application requirements [25], [55]. Since the idea of software platforms is inspired by the mechanical domain, this is not surprising and the logical difference is a shift from physical artifacts towards software artifacts. However, the term “software platform” is constantly used in conjunction with the term “hardware platform,” which is considered as the technical enabler for standardizing software artifacts [25], [55]. So, the software platform includes all reusable software artifacts, while the hardware platform needs to fulfill all architectural constraints to enable the reuse of software and hardware components. As a result, software and hardware platform are always closely connected to each other, with the combination of both being called a “system platform” [25], [55], [85]. The “electrics/electronics platform” or “architecture platform” extends this idea by emphasizing a combined platform for modern systems that

TABLE III: Terms we extracted from each paper.

		Architecture				Platform				Variability	Variety	Component		Product Line	
Reference	Domain	Universal	Product	Hardware	Software	Universal	Product	Hardware	Software	Universal	Product	Universal	Modular	Universal	Software
Benavides et al. (2010) [7]	SW	○	○	○	○	○	○	○	○	○	○	○	○	○	◐
Broy et al. (2007) [12]	SW	○	○	●	●	○	○	○	○	○	○	○	○	○	○
Classen et al. (2013) [16]	SW	○	○	○	○	○	○	○	○	○	○	○	○	○	●
Clements and Northrop (2001) [17]	SW	○	○	○	○	○	○	○	○	○	○	○	○	●	●
Eklund and Gustavsson (2013) [23]	SW	○	○	○	○	○	○	○	○	○	○	○	○	○	●
Ferrari and Sangiovanni-Vincentelli (1999) [25]	SW	●	○	○	○	○	○	●	●	○	○	○	○	○	○
Galster et al. (2014) [32]	SW	○	○	○	○	○	○	○	○	◐	○	○	○	●	○
Oreizy et al. (1998) [79]	SW	○	○	○	◐	○	○	○	○	○	○	●	○	○	○
Perry and Wolf (1992) [82]	SW	●	○	○	◐	○	○	○	○	○	○	○	○	○	○
Queiroz and Braga (2014) [88]	SW	○	○	○	○	○	○	○	○	○	○	○	○	○	●
Schmid and John (2004) [97]	SW	○	○	○	○	○	○	○	○	◐	○	○	○	◐	○
Shaw et al. (1995) [100]	SW	○	○	○	●	○	○	○	○	○	○	○	○	○	○
Thüm et al. (2014) [107]	SW	○	○	○	○	○	○	○	○	○	○	○	○	○	◐
Thiel et al. (2009) [106]	SW	○	○	○	○	○	○	○	○	○	○	○	○	○	●
van Gurp et al. (2001) [111]	SW	○	○	○	○	○	○	○	○	●	○	○	○	○	●
Voelter and Groher (2007) [113]	SW	○	○	○	○	○	○	○	○	●	○	○	○	○	○
Browning (2001) [10]	E/E	◐	◐	○	○	○	○	○	○	○	○	○	○	○	○
Flores et al. (2012) [29]	E/E	○	○	○	○	○	○	○	○	○	○	●	○	○	○
Gleirscher et al. (2014) [33]	E/E	○	○	○	○	●	○	○	○	○	○	○	●	○	○
Huang and Kusiak (1998) [46]	E/E	○	○	○	○	○	○	○	○	○	○	○	◐	○	○
Jaensch et al. (2010) [48]	E/E	○	○	○	○	●	○	○	○	○	○	○	○	○	○
Keutzer et al. (2000) [55]	E/E	●	○	○	○	○	○	●	●	○	○	◐	○	○	○
Pohl et al. (2005) [85]	E/E	○	○	○	○	●	○	○	●	○	○	○	○	●	●
Sangiovanni-Vincentelli and Martin (2001) [96]	E/E	○	○	○	○	●	○	○	○	○	○	○	○	○	○
Fahl et al. (2019) [24]	ME	○	○	○	○	○	○	○	○	○	○	○	○	●	○
Fisher et al. (1995) [27]	ME	○	○	○	○	○	○	○	○	○	◐	○	○	○	○
Fisher et al. (1999) [28]	ME	○	○	○	○	○	●	○	○	○	◐	○	○	○	○
Gershenson and Zhang (2003) [47]	ME	○	◐	○	○	●	○	○	○	○	◐	○	◐	○	○
Messac et al. (2002) [73]	ME	○	○	○	○	○	◐	○	○	○	○	○	○	○	○
Meyer and Lehnerd (1997) [75]	ME	○	○	○	○	●	●	○	○	○	○	○	○	○	○
Muffatto (1999) [77]	ME	○	○	○	○	◐	●	○	○	○	○	○	◐	○	○
Simpson et al. (2001) [102]	ME	○	○	○	○	○	●	○	○	○	●	○	○	○	○
Ulrich (1995) [108]	ME	○	●	○	○	○	●	○	○	○	●	●	●	○	○
Wilhelm (1997) [116]	ME	○	○	○	○	○	○	○	○	○	○	○	●	○	○
Henderson and Clark (1990) [41]	OD	○	○	○	○	○	○	○	○	○	○	●	○	○	○
Koufteros et al. (2002) [61]	OD	○	○	○	○	◐	○	○	○	○	○	○	○	○	○
Langlois and Robertson (1992) [66]	OD	○	○	○	○	○	○	○	○	○	○	○	◐	○	○
Mikkola and Gassmann (2003) [76]	OD	○	◐	○	○	○	○	○	○	○	○	◐	○	○	○
Robertson and Ulrich (1998) [92]	OD	○	○	○	○	○	◐	○	○	○	○	●	○	○	○
Sanchez (1996) [94]	OD	○	◐	○	○	○	○	○	○	○	○	○	○	○	○

●: Primary definition available; ◐: Secondary definition available; ○: No definition available
SW: Software Engineering; E/E: Electrics/Electronics Engineering;
ME: Mechanical Engineering; OD: Other Domain

incorporate software, hardware, mechanics, electrics/electronics, and their interconnections [25], [48], [85], [96].

Product Line. The term "product line" is mostly used in the software-engineering domain, typically referred to as a "software product line." In fact, we included no paper from mechanical engineering that referred to a product line, despite the fundamental definition of product-line engineering addressing products and their variability in general. Product-line engineering is focused around exploiting reuse potential among products within an organization by identifying commonalities between the products and systematizing their variabilities (i.e., configurable features) [17], [32], [85], [97]. The term "product line" typically refers to a product family, wherein individual products show a high overlap of mandatory or commodity features [17], [85]. In the context of product-generation engineering, the definition of a product line may also apply to a series of successive product generations [24].

"Software product-line engineering" builds upon these definitions, aiming to develop complex software products as a "software product line" [16], [17], [111]. The term "software product line" is commonly defined as "a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way" [7], [17], [23], [88], [106]. Thüm et al. [107] take this definition to the source-code level and characterize a software product line as a software product family sharing a common code base, with the individual products differing in the implemented features. In each case, the focus of software product lines is on reducing and managing variability by achieving reuse and synergy potentials, thereby leading to savings in development and maintenance costs [7], [17], [85], [111].

Variability. Regarding variability and variant management, we found different terms, but each of them was typically utilized in only one distinct domain. Within software engineering, the term "variability" is often used to describe the ability of developers to customize (a.k.a., configure) a platform depending on specific requirements [32], [111]. The respective variability management primarily pertains to software product-line engineering and involves defining parameters as well as variation points within a product line and managing these throughout the entire life cycle [32], [97], [111], [113]. It is worth noting that variability management focuses on planning and systematizing software changes to create variants, taking into account possible reuse and standardization. Thus, variability is considered essential to enable the adaptability of a software product line to customer preferences and requirements while facilitating reuse synergies. However, Galster et al. [32] note that variability is not exclusive to software product lines, but impacts all kinds of software systems and their development processes. As a result, variability represents an inherently existing challenge for engineering complex software systems.

While software engineering focuses on the term "variability" to describe variant-management activities and challenges, mechanical engineering papers mostly refer to the term "product variety." The term "product variety" addresses the diversity of product families that results from the demand for highly configurable, mass-produced products, also referred to as mass customization [28], [47], [108]. In this context, variety also involves distinguishing between the breadth of the product portfolio and the frequency of product-generation changes [27], [28], [108]. For the automotive industry, the breadth is typically further divided into subtypes of fundamental variety (e.g., different platforms, models, body styles) and peripheral variety (e.g., options, packages, equipment) [27]. In contrast to "variability," most papers establish a direct link between "product variety" and the expenses as well as challenges of excessive product diversity—highlighting the need for effective variant management [27], [102], [108].

Overall, we identified a distinct conceptual difference of variant-management terminologies between the domains of software engineering and mechanical engineering. Software engineering is consistently using the term "variability," whereas mechanical engineering mostly refers to the term "product variety." Nonetheless, the definitions are highly similar to identical.

Component. We found papers from every domain that define the term "component." Irrespective of the domain, a "component" is understood as a distinctly identifiable unit or part fulfilling a particular functionality [25], [29], [41], [55], [79], [92], [108]. Depending on that functionality, components may involve hardware, software, or mechanical elements [29], [41], [76], [108]. To achieve cross-product synergies, groups of components are commonly consolidated into modules [66], [78], [116]. Modules are characterized by standardized interfaces, allowing for independent development and manufacturing as well as for deployment in multiple different products [46], [47], [78], [108], [116].

RQ$_1$ & RQ$_2$: Overview of Terminologies

We found domain-specific definitions for the terms "architecture," "platform," and "variability"/"variant management." In contrast, the term "component" is consistently defined and used across domains. The term "(software) product line" is predominantly used within software engineering. While this is a clear list of key terms (RQ$_1$), their definitions across domains vary (RQ$_2$).

VI. Discussion (RQ$_3$)

In this section, we discuss our findings regarding existing definitions, focusing on the differences and commonalities across domains. We provide an overview of the terms we extracted and analyzed in Figure 7. By comparing the definitions, we derive practical challenges that can occur within cross-sectional application domains. In the end, we provide a framework in which we aimed to introduce a cohesive definition and overview of the terms and their relations. Due to our experiences, we use the automotive industry for illustrative examples.

Architecture. The term "architecture" is defined and used across domains in a similar fashion. However, we also found that the term is typically connected to some other term that is applied strictly in one domain. Specifically, mechanical engineering typically refers to the "product architecture" whereas software engineering refers to "hardware/software architecture." Consequently, different terms exist for closely related concepts, which can lead to communication issues in practice; especially within cross-functional teams and processes. In fact, we could not identify a cohesive, cross-disciplinary understanding of architecture in the literature, which would enhance mutual comprehension in interdisciplinary projects.

Platform. The term "product platform" is widely used in mechanical engineering, particularly in the automotive industry. In this industry, it denotes a consolidated set of technical components that are used across multiple products within a product family. While we have encountered definitions that extend the platform concept beyond technical components to processes and relationships, we have not found an explicit association of product platforms to software. Instead, the term "software platform" has been established within software engineering, which follows the same idea as a product platform, albeit focusing on the reuse of software artifacts. Given the intricate interactions among hardware, software, and mechanical components in modern vehicles, a distinction between software, hardware, and product platforms may lead to communication issues and may impede synergies. Recognizing this issue, recent papers have explored the development

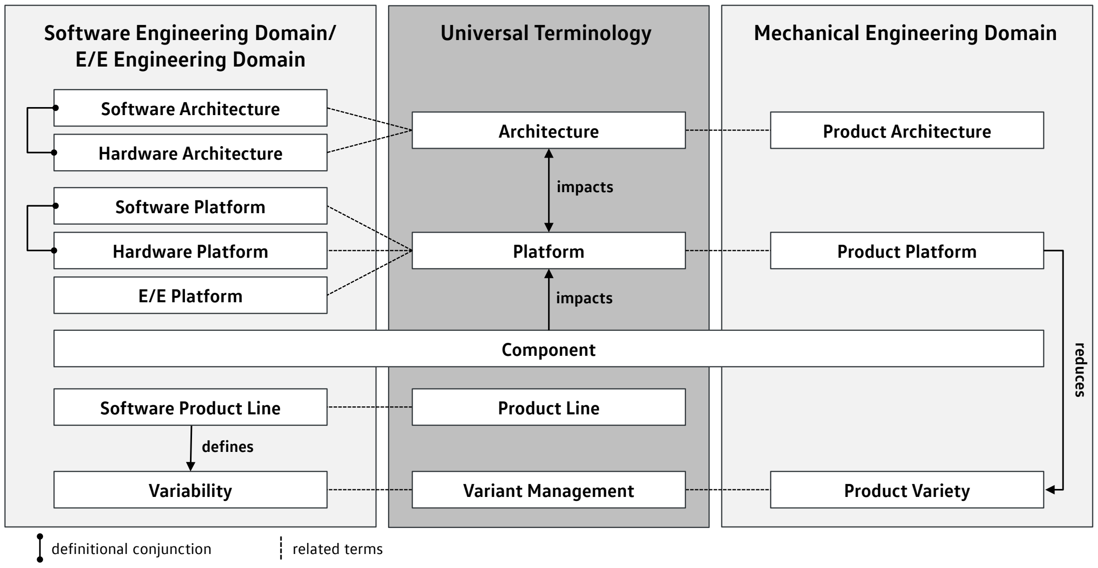

Fig. 7: Conceptional model of key terms of automotive product-structuring concepts.

of more integrated concepts, such as "electrics/electronics platforms" or "architecture platforms." These emphasize an overarching electrics/electronics vehicle architecture that efficiently combines hardware and software artifacts across different vehicle models. However, we found no reports that such an integrated concept is already applied in practice.

Product Line. Product-line engineering, and especially software product-line engineering, are widely recognized within software engineering. However, the concept seems to have low traction on mechanical engineering. Comparing the definitions of "product line" and "product platform" reveals a consistent underlying concept, with domain-specific properties. The common idea is to standardize a set of components or artifacts across a product family, aiming to facilitate reuse and synergies to achieve a systematic management of complexity. In essence, product-line engineering aims to establish a systematically managed and configurable platform, closely connecting both terms. Within the automotive industry, synergies and reuse potentials have traditionally been achieved through mechanically oriented platforms and modularization strategies. However, the increasing digitization and the resulting relevance of software highlight the need for an growing focus on hardware and software components as key artifacts to reduce the overall complexity. Such considerations have already been successfully studied in software engineering through software product-line engineering, but we have not identified such integrated concepts within vehicle portfolios. Based on our findings, we argue that integrating software-complexity management approaches like software product-line engineering into existing mechanically-oriented platform engineering poses a practical and conceptual challenge. Particularly, it seems that there are missing interconnections between closely related concepts in the literature.

Variability. As we outlined in Section V, different terminologies are used for variant management across domains. In software engineering, the discourse revolves around the term "variability" and, in turn, "variability management." In contrast, mechanical engineering uses the term "product variety," especially within the context of conventional automotive development. Fundamentally, both terminologies share the same core: Both describe the ability of technical products to be customized and configured to derive variants that fulfill a customer's requirements. Nevertheless, both terminologies are strictly used within their respective domain; we did no observe any overlaps. Again, this highlights a tendency of missing cross-domain knowledge, which may partly be caused by the different terminologies. So, we argue that further cross-domain research is needed to build on each domains' expertise, avoid costly redevelopment of the same concepts, and support practitioners with a more integrated understanding.

Component. The term "component" appears to be a prime example of a cross-domain understanding, as we have encountered the term in all examined domains with a consistent definition. Consequently, we perceive that this term is well-aligned between researchers from different domains. Still, this does not imply that the term is ideally defined within each domain. For instance, the boundaries of a software component are typically more vague and up for debate among engineers compared to a mechanical component.

Summary. Our findings indicate that there are no consistently used cross-domain definitions for key terms used for automotive product-structuring concepts (cf. Figure 7). For instance, software-engineering researchers consistently use the terms hardware or software architecture and hardware or software platform. In contrast, mechanical-engineering researchers refer to product architecture and product platform. Essentially, each domain is adjusting the definitions of core concepts to their domain specifics. By itself, such adjustments are not a problem. However, for engineering modern systems that incorporate layers from various domains, these adjustments can cause confusions, miscommunication, and potentially wrong assumptions. By introducing the

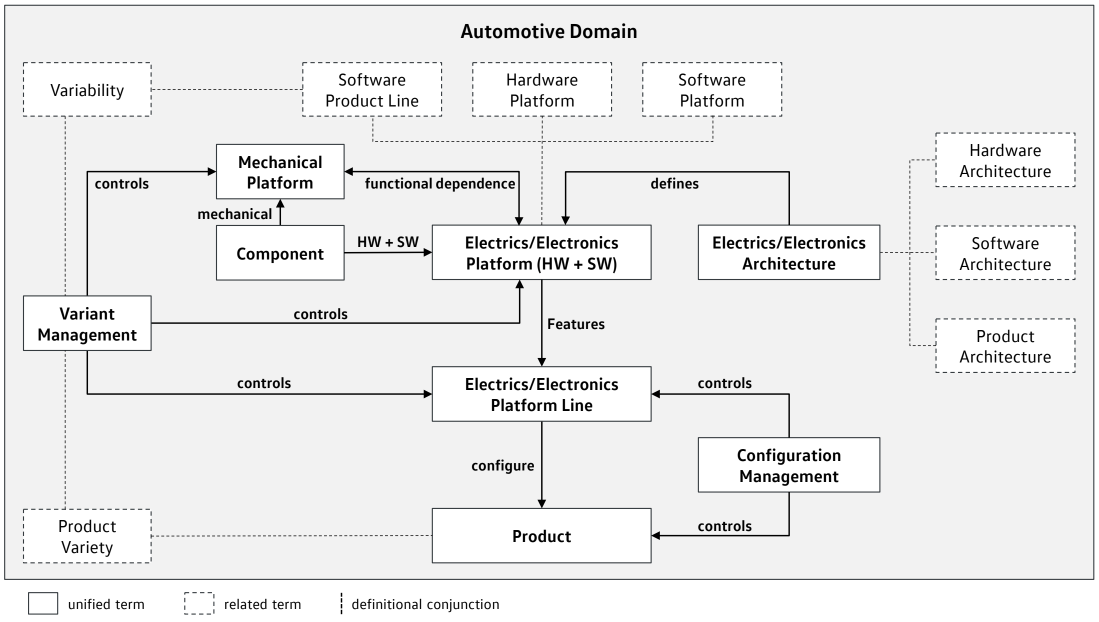

Fig. 8: Our conceptual framework of product-structuring concepts.

concept of electrics/electronics platforms, some researchers have attempted to unify the terminological understanding. Yet, its use in research and practice is lacking to date.

Another issue arises from the use of the term product line: Despite high similarities within the definitions of platform and product line, we found no overarching terminology that clearly distinguishes or standardizes these terms. Instead, we found no paper from mechanical engineering utilizing the term product line at all. We consider such terminological vagueness a challenge for automotive practitioners to integrate software-engineering methods like software product-line engineering into existing automotive platform strategies.

We observed a similar situation regarding variant management: Distinct terms are employed across domains, but at their core they describe the same concept. Both, the term variability from software engineering and the term product variety from mechanical engineering characterize product diversity and customized product variants. Nevertheless, we found no terminological framework that defines the similarities and relations between these terms.

RQ3: Challenges and Implications

Software engineering and mechanical engineering lack a defined and standardized cross-domain terminology. This can confuse researchers and practitioners, motivating future research towards an integrated framework.

Conceptual Framework. Building on our findings, we derived the conceptual framework that we illustrate in Figure 8. Specifically, we structured and defined the key terms related to automotive product-structuring concepts we summarize in Figure 7. This framework aims to support practitioners, particularly from the automotive industry, in successfully adopting and integrating engineering methods from different domains to effectively implement an overarching platform approach. Please note that we focus on the automotive industry, but the terms and our framework can be transferred across other industries and domains, too.

Within our framework, the terms “electrics/electronics platform,” “electrics/electronics platform line,” and “product” are central, each representing a different level within an automotive product portfolio. First, the “electrics/electronics platform” describes the entirety of reusable artifacts that encompasses all hardware and software components, providing a cross-vehicle basis within a product portfolio. Second, the “electrics/electronics platform line” refers only to a subset of the platform sharing identical components. This level is optional in different contexts, but from our experiences, it can be very helpful, for instance, to distinguish between different models. Lastly, the “product” refers to a single vehicle that is derived from the “electrics/electronics platform” and customized to customer requirements.

At the top level, the “electrics/electronics architecture” defines all connections between the individual hardware and software components within the “electrics/electronics platform.” Consequently, this architecture represents the connections between, for example, control units, sensors, and actuators. Moreover, the “electrics/electronics platform” and the “mechanical platform” are interdependent. Electrics/electronics components (a combination of hardware and software components [43]) can impact the mechanical platform, and mechanical components can impact the electrics/electronics platform—which, in turn, is represented as variability. To manage this variability, “variant management” controls the electrics/electronics platform, the number of electrics/electronics platform lines, and the scope within the mechanical platform. We distinguish variant management from configuration management, since it typically aims to achieve an optimal number of variants within the product portfolio. In contrast, the “configuration management” is concerned with configuring and deriving an individual product from the electrics/electronics platform based on the variability and thereby variants defined in the variant management. Within our framework, we refer to "mechanical platform" instead of product platform to emphasize the dependency on mechanical components and to avoid misunderstandings related to similar terms from the literature. In our framework, the mechanical platform comprises only components that are not part of the electrics/electronics platform, such as brake disks or suspension springs.

VII. Threats to Validity

We recognize that the validity of our mapping study may be threatened. First, the level of detail regarding information we needed varied among the papers. Some contain relevant definitions and descriptions in great detail, others comprise no relevant information or miss important pieces. In this regard, we implemented quality checks based on our selection criteria to mitigate such issues and leveraged our practical experiences to contextualize the results. Inherently, all analyses and interpretations of the data are subject to our interpretation, which threatens the construct and internal validity because we may have misunderstood constructs or details were missing. To reduce subjectivity, we rigorously followed our methodology, particularly the search strategy outlined in Section IV-B with two independent analysts and cross-validations among all authors. However, we cannot guarantee that we did not misinterpret some pieces of information.

Regarding the external validity of our study, we first acknowledge that our search strategy does neither encompass all papers related to the domains we list in Table III nor to all product-structuring concepts that exist in these. Our selection of relevant papers is based on predefined data sources, and thus limited to the available literature in these. This may have introduced sampling biases that threatens the external validity of our mapping study. Moreover, we narrowed down the number of papers to a final selection of 40 from multiple databases. This reduction may have increased the likelihood of incorrect classifications, due to the smaller sample size. To address these threats, we involved multiple researchers in the literature analysis, performed a thorough exploration of various databases (i.e., IEEE, Scopus, ACM), and employed snowballing. Moreover, we documented each stage of our process to facilitate transparency and reproducibility.

VIII. Conclusion

In this article, we presented a systematic mapping study on existing terminologies used for automotive product-structuring concepts. We aimed to provide an overview of established terms, their definitions, their differences, and their relations within as well as across domains. For this purpose, we analyzed 40 highly-cited papers, from which we extracted relevant terms and definitions. Our results indicate that most key terms used for describing automotive product-structuring concepts have distinct domain-specific definitions. Notably, "platform" and "architecture" are each specified differently in software engineering and mechanical engineering, despite their conceptually equivalent meaning across these domains. The term "product line" exhibits significant overlap with the term "platform," but is employed exclusively in software engineering. Regarding variant management, the terms "variability" and "product variety" describe parallel concepts, yet are used exclusively within specific domains. To tackle these conceptual differences, we proposed a first conceptual model that defines distinct terms and shows how these relate to each other. We hope that our mapping study and framework offer assistance for practitioners implementing product-structuring concepts, and for researchers to facilitate cross-domain research.

We have placed strong emphasis on mechanical engineering, electrics/electronics engineering, and software engineering as key domains in automotive development. Future research could build on this foundation and expand its scope to include other industries and domains, such as life-cycle assessment, security by design, or artificial intelligence. This way, our framework could be refined and expanded in the future. Moreover, we see the need to evaluate the individual product-structuring concepts as well as our framework in practice to understand their potential pros and limitations.

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

Appendix

In Table IV and Table V, we provide an overview of the studies we included from our initial mapping study [118] and from our new literature search, respectively.

References

[1] H.-B. Abel, H.-J. Blume, L. Brabetz, M. Broy, S. Fürst, L. Ganzelmeier, J. Helbig, G. Heyen, M. Jipp, G. Kasties, P. Knoll, O. Krieger, R. Lachmayer, K. Lemmer, W. Pfaff, T. Scharnhorst, and G. Schneider, *Elektrik/Elektronik/Software*. Springer, 2016.

[2] A. Albers, N. Bursac, and E. Wintergerst, "Product generation development - importance and challenges from a design research perspective," in *International Conference on Theoretical Mechanics and Applied Mechanics (TMAM)*. INASE, 2015.

[3] A. Albers, J. Fahl, T. Hirschter, M. Endl, R. Ewert, and S. Rapp, "Model of pge – product generation engineering by the example of autonomous driving," *Procedia CIRP*, vol. 91, 2020.

[4] A. Albers, S. Rapp, J. Fahl, T. Hirschter, S. Revfi, M. Schulz, T. Stürmlinger, and M. Spadinger, "Proposing a generalized description of variations in different types of systems by the model of pge – product generation engineering," *International Design Conference (DESIGN)*, 2020.

[5] J. Axelsson, "A transformation-based model of evolutionary architecting for embedded system product lines," in *International Workshop on Model Based Architecting and Construction of Embedded Systems (ACES-MB)*. CEUR-WS.org, 2010.

[6] S. Baumgart, X. Zhang, J. Fröberg, and S. Punnekkat, "Variability management in product lines of safety critical embedded systems," in *International Conference on Embedded Systems (ICES)*. ACM, 2014.

[7] D. Benavides, S. Segura, and A. Ruiz-Cortés, "Automated analysis of feature models 20 years later: A literature review," *Information Systems*, vol. 35, no. 6, pp. 615–636, 2010.

[8] D. Bilic, E. Brosse, A. Sadovykh, D. Truscan, H. Bruneliere, and U. Ryssel, "An integrated model-based tool chain for managing variability in complex system design," in *International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, 2019.

[9] C. Brink, E. Kamsties, M. Peters, and S. Sachweh, "On hardware variability and the relation to software variability," in *Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2014.

[10] T. R. Browning, "Applying the design structure matrix to system decomposition and integration problems: a review and new directions," *IEEE Transactions on Engineering Management*, vol. 48, pp. 292–306, 2001.

TABLE IV: Studies from our initial mapping study [118] that we included in this work.

ID	Authors	Year	Venue	Title	Source
01	Eklund and Gustavsson [23]	2013	SCP	Architecting Automotive Product Lines: Industrial Practice	ACM
02	Fahl et al. [24]	2019	ISSE	Functional Concepts in the model of PGE – Product Generation Engineering by the Example of Automotive Product Development	IEEE
03	Flores et al. [29]	2012	SPLC	Mega-Scale Product Line Engineering at General Motors	ACM
04	Gleirscher et al. [33]	2014	IWSPM	A Model-Based Approach to Innovation Management of Automotive Control Systems	IEEE
05	Jaensch et al. [48]	2010	GI	Transfer von Prozessen des Software-Produktlinien Engineering in die Elektrik/Elektronik-Architekturentwicklung von Fahrzeugen	SB
06	Queiroz and Braga [88]	2014	SEKE	A Critical Embedded System Product Line Model-based Approach	Scopus
07	Thiel et al. [106]	2009	SAE	Software Product Lines in Automotive Systems Engineering	Scopus

ACM: ACM Digital Library; IEEE: IEEE Xplore; SB: Snowballing

TABLE V: Studies from our new literature search that we included in this work.

ID	Authors	Year	Venue	Title	Source
01	Benavides et al. [7]	2010	ISJ	Automated Analysis of Feature Models 20 Years Later	Scopus
02	Browning [10]	2001	TEM	Applying the Design Structure Matrix to System Decomposition and Integration Problems	IEEE
03	Broy et al. [12]	2007	PROC	Engineering Automotive Software	SB
04	Classen et al. [16]	2013	TSE	A Classification and Survey of Analysis Strategies for Software Product Lines	ACM
05	Clements and Northrop [17]	2002	BC	Software Product Lines	SB
06	Ferrari and Sangiovanni-Vincentelli [25]	1999	ICCD	System Design: Traditional Concepts and New Paradigms	SB
07	Fisher et al. [27]	1995	BC	Strategies for Product Variety: Lessons from the Auto Industry	SB
08	Fisher et al. [28]	1999	MANS	Component Sharing in the Management of Product Variety	ACM
09	Galster et al. [32]	2014	TSE	Variability in Software Systems— A Systematic Literature Review	IEEE
10	J. K. Gershenson and Zhang [47]	2003	JED	Product modularity: definitions and benefits	Scopus
11	Henderson and Clark [41]	1990	ASQ	Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms	SB
12	Huang and Kusiak [46]	1998	SMC	Modularity in Design of Products and Systems	SB
13	Keutzer et al. [55]	2000	TCAD	System-Level Design: Orthogonalization of Concerns and Platform-Based Design	IEEE
14	Koufteros et al. [61]	2002	JOM	Integrated product development practices and competitive capabilities	Scopus
15	Langlois and Robertson [66]	1992	RP	Networks and innovation in a modular system	SB
16	Messac et al. [73]	2002	JMD	Introduction of a Product Family Penalty Function Using Physical Programming	Scopus
17	Meyer and Lehnerd [75]	1997	BC	The Power of Product Platforms	SB
18	Mikkola and Gassmann [76]	2003	TEM	Managing Modularity of Product Architectures: Toward an Integrated Theory	IEEE
19	Muffatto [77]	1999	IJPE	Introducing a platform strategy in product development	SB
20	Oreizy et al. [79]	1998	ICSE	Architecture-Based Runtime Software Evolution	IEEE
21	Perry and Wolf [82]	1992	SEN	Foundations for the Study of Software Architecture	SB
22	Pohl et al. [85]	2005	BC	Software Product Line Engineering	SB
23	Robertson and Ulrich [92]	1998	SMR	Planning for Product Platforms	SB
24	Sanchez [94]	1996	EMJ	Strategic Product Creation: Managing New Interactions of Technology, Markets, and Organizations	SB
25	Sangiovanni-Vincentelli and Martin [96]	2001	D&T	Platform-Based Design and Software Design Methodology for Embedded Systems	IEEE
26	Schmid and John [97]	2004	SCP	A customizable approach to full lifecycle variability management	SB
27	Shaw et al. [100]	1995	TSE	Abstractions for Software Architecture and Tools to Support Them	IEEE
28	Simpson et al. [102]	2001	RED	Product platform design: method and application	SB
29	Thüm et al. [107]	2014	CSUR	A Classification and Survey of Analysis Strategies for Software Product Lines	ACM
30	Ulrich [108]	1995	RP	The role of product architecture in the manufacturing firm	SB
31	van Gurp et al. [111]	2001	ICSA	On the Notion of Variability in Software Product Lines	SB
32	Voelter and Groher [113]	2007	SPLC	Product Line Implementation using Aspect-Oriented and Model-Driven Software Development	IEEE
33	Wilhelm [116]	1997	BC	Platform and modular concept at Volkswagen - their effect on the assembly process	SB

ACM: ACM Digital Library; IEEE: IEEE Xplore; SB: Snowballing, BC: Book Chapter

[11] M. Broy, "Challenges in automotive software engineering," in *International Conference on Software Engineering (ICSE)*. ACM, 2006.

[12] M. Broy, I. H. Kruger, A. Pretschner, and C. Salzmann, "Engineering automotive software," *IEEE*, vol. 95, no. 2, pp. 356–373, 2007.

[13] A. Bucaioni and P. Pelliccione, "Technical architectures for automotive systems," in *International Conference on Software Architecture (ICSA)*. IEEE, 2020, pp. 46–57.

[14] H. Bucher, K. Neubauer, and J. Becker, "Automated assessment of e/e-architecture variants using an integrated model- and simulation-based approach," in *World Congress Experience (WCX)*. SAE International, 2019.

[15] C. Buckl, A. Camek, G. Kainz, C. Simon, L. Mercep, H. Stähle, and A. Knoll, "The software car: Building ict architectures for future electric vehicles," in *International Electric Vehicle Conference (IEVC)*. IEEE, 2012.

[16] A. Classen, M. Cordy, P.-Y. Schobbens, P. Heymans, A. Legay, and J.-F. Raskin, "Featured transition systems: Foundations for verifying variability-intensive systems and their application to ltl model checking," *IEEE Transactions on Software Engineering*, vol. 39, no. 8, pp. 1069–1089, 2013.

[17] P. C. Clements and L. M. Northrop, *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2001.

[18] B. Cool, C. Knieke, A. Rausch, M. Schindler, A. Strasser, M. Vogel, O. Brox, and S. Jauns-Seyfried, "From product architectures to a managed automotive software product line architecture," in *Symposium on Applied Computing (SAC)*. ACM, 2016.

[19] Y. Dajsuren and M. v. den Brand, "Automotive software engineering: Past, present, and future," in *Automotive Systems and Software Engineering*. Springer, 2019.

[20] M. J. B. de Sousa, L. F. G. Gonzalez, E. M. Ferdinando, and J. F. Borin, "Over-the-air firmware update for iot devices on the wild," *Internet of Things*, vol. 19, 2022.

[21] O. L. de Weck, E. S. Suh, and D. Chang, "Product family and platform portfolio optimization," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC)*. ASME, 2003.

[22] M. Eigner, W. Koch, and C. Muggeo, *Modellbasierter Entwicklungsprozess cybertronischer Systeme: Der PLM-unterstützte Referen-*

zentwicklungsprozess für Produkte und Produktionssysteme. Springer, 2017.

[23] U. Eklund and H. Gustavsson, “Architecting automotive product lines: Industrial practice,” *Science of Computer Programming*, vol. 78, no. 12, 2013.

[24] J. Fahl, T. Hirschter, J. Kamp, M. Endl, and A. Albers, “Functional concepts in the model of pge – product generation engineering by the example of automotive product development,” in *International Symposium on Systems Engineering (ISSE)*. IEEE, 2019.

[25] A. Ferrari and A. Sangiovanni-Vincentelli, “System design: traditional concepts and new paradigms,” in *International Conference on Computer Design: VLSI in Computers and Processors (Cat. No.99CB37040)*. IEEE, 1999, pp. 2–12.

[26] S. Fischer, L. Linsbauer, R. E. Lopez-Herrejon, A. Egyed, and R. Ramler, “Bridging the gap between software variability and system variant management: Experiences from an industrial machinery product line,” in *Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2015.

[27] M. Fisher, A. Jain, and J. P. Macduffie, “Strategies for Product Variety: Lessons from the Auto Industry,” in *Redesigning the Firm*. Oxford University Press, 1995.

[28] M. L. Fisher, K. Ramdas, and K. T. Ulrich, “Component sharing in the management of product variety: a study of automotive braking systems,” *Management Science*, vol. 45, pp. 297–315, 1999.

[29] R. Flores, C. Krueger, and P. Clements, “Mega-scale product line engineering at general motors,” in *International Software Product Line Conference (SPLC)*. ACM, 2012.

[30] A. Frank and E. Brenner, “Model-based variability management for complex embedded networks,” in *International Multi-Conference on Computing in the Global Information Technology (ICCGI)*. IEEE, 2010.

[31] C. Frank, L. Holsten, T. Şahin, and T. Vietor, “How to manage vehicle platform variants? a method to assess platform variance through competitive analysis,” *Procedia CIRP*, vol. 109, 2022.

[32] M. Galster, D. Weyns, D. Tofan, B. Michalik, and P. Avgeriou, “Variability in software systems—a systematic literature review,” *IEEE Transactions on Software Engineering*, vol. 40, no. 3, pp. 282–306, 2014.

[33] M. Gleirscher, A. Vogelsang, and S. Fuhrmann, “A model-based approach to innovation management of automotive control systems,” in *International Workshop on Software Product Management (IWSPM)*. IEEE, 2014.

[34] S. Graf, M. Glaß, J. Teich, and C. Lauer, “Design space exploration for automotive e/e architecture component platforms,” in *Euromicro Conference on Digital System Design (DSD)*. IEEE, 2014.

[35] S. Graf, S. Reinhart, M. Glaß, J. Teich, and D. Platte, “Robust design of e/e architecture component platforms,” in *Design Automation Conference (DAC)*. IEEE, 2015.

[36] H. Guissouma, C. P. Hohl, F. Lesniak, M. Schindewolf, J. Becker, and E. Sax, “Lifecycle management of automotive safety-critical over the air updates: A systems approach,” *IEEE Access*, pp. 57 696–57 717, 2022.

[37] H. Gustavsson and J. Axelsson, “Evaluating flexibility in embedded automotive product lines using real options,” in *International Software Product Line Conference (SPLC)*. IEEE, 2008.

[38] S. Halder, A. Ghosal, and M. Conti, “Secure over-the-air software updates in connected vehicles: A survey,” *Computer Networks*, vol. 178, 2020.

[39] K. Hayashi and M. Aoyama, “A multiple product line development method based on variability structure analysis,” in *International Systems and Software Product Line Conference (SPLC)*. ACM, 2018.

[40] M. Hayat and H. Winkler, “Exploring the basic features and challenges of traditional product lifecycle management systems,” in *International Conference on Industrial Engineering and Engineering Management (IEEM)*. IEEE, 2022.

[41] R. M. Henderson and K. B. Clark, “Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms,” *Administrative Science Quarterly*, vol. 35, no. 1, pp. 9–30, 1990.

[42] H. Hick, K. Küpper, and H. Sorger, *Systems Engineering for Automotive Powertrain Development*. Springer, 2021.

[43] L. Holsten, C. Frank, J. Krüger, and T. Leich, “Electrics/electronics platforms in the automotive industry: Challenges and directions for variant-rich systems engineering,” in *International Working Conference on Variability Modelling of Software-Intensive Systems*. ACM, 2023.

[44] L. Holsten, J. Krüger, and T. Leich, “Insights into transitioning towards electrics/electronics platform management in the automotive industry,” in *International Conference on the Foundations of Software Engineering*, ser. FSE 2024. ACM, 2024, p. 161–172.

[45] K. Hölttä-Otto, “Modular product platform design,” Ph.D. dissertation, Helsinki University of Technology, 2005.

[46] C.-C. Huang and A. Kusiak, “Modularity in design of products and systems,” *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 28, no. 1, pp. 66–77, 1998.

[47] G. J. P. J. K. Gershenson and Y. Zhang, “Product modularity: Definitions and benefits,” *Journal of Engineering Design*, vol. 14, no. 3, pp. 295–313, 2003.

[48] M. Jaensch, B. Hedenetz, M. Conrath, and K. D. Müller-Glaser, “Transfer von prozessen des software-produktlinien engineering in die elektrik/elektronik- architekturentwicklung von fahrzeugen,” in *INFORMATIK*. GI, 2010.

[49] C. Jiacheng, Z. Haibo, Z. Ning, Y. Peng, G. Lin, and S. X. Sherman, “Software defined internet of vehicles: Architecture, challenges and solutions,” *Journal of Communications and Information Networks*, vol. 1, no. 1, 2016.

[50] E.-Y. Kang, D. Mu, L. Huang, and Q. Lan, “Verification and validation of a cyber-physical system in the automotive domain,” in *International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, 2017.

[51] S. Karnouskos, “Cyber-physical systems in the smartgrid,” in *International Conference on Industrial Informatics (INDIN)*. IEEE, 2011.

[52] S. Kato and N. Yamaguchi, “Variation management for software product lines with cumulative coverage of feature interactions,” in *International Software Product Line Conference (SPLC)*. IEEE, 2011.

[53] A. Kenner, R. May, J. Krüger, G. Saake, and T. Leich, “Safety, security, and configurable software systems: A systematic mapping study,” in *International Systems and Software Product Line Conference (SPLC)*. ACM, 2021.

[54] K. Kerliu, A. Ross, G. Tao, Z. Yun, Z. Shi, S. Han, and S. Zhou, “Secure over-the-air firmware updates for sensor networks,” in *International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW)*. IEEE, 2019.

[55] K. Keutzer, A. Newton, J. Rabaey, and A. Sangiovanni-Vincentelli, “System-level design: orthogonalization of concerns and platform-based design,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 12, pp. 1523–1543, 2000.

[56] B. Kitchenham, “Evidence-based software engineering and systematic literature reviews,” in *International Conference on Product Focused Software Process Improvement (PROFES)*. Springer, 2006.

[57] B. A. Kitchenham, D. Budgen, and O. Pearl Brereton, “Using mapping studies as the basis for further research – a participant-observer case study,” *Information and Software Technology*, vol. 53, 2011.

[58] B. A. Kitchenham and S. Charters, “Guidelines for performing systematic literature reviews in software engineering,” Keele University and Durham University, Tech. Rep. EBSE 2007-001, 2007.

[59] C. Knieke, A. Rausch, M. Schindler, A. Strasser, and M. Vogel, “Managed evolution of automotive software product line architectures: A systematic literature study,” *Electronics*, vol. 11, 2022.

[60] C. F. J. König, G. Meisl, N. Balcu, B. Vosseler, H. Hörmann, J. Höll, and V. Fäßler, “Engineering of cyber-physical systems in the automotive context: Case study of a range prediction assistant,” in *International Symposium on Leveraging Applications of Formal Methods (ISoLA)*. Springer, 2018.

[61] X. A. Koufteros, M. A. Vonderembse, and W. J. Doll, “Integrated product development practices and competitive capabilities: the effects of uncertainty, equivocality, and platform strategy,” *Journal of Operations Management*, vol. 20, no. 4, pp. 331–355, 2002.

[62] J. Krüger, “Understanding the Re-Engineering of Variant-Rich Systems: An Empirical Work on Economics, Knowledge, Traceability, and Practices,” Ph.D. dissertation, Otto-von-Guericke University Magdeburg, 2021.

[63] J. Krüger and T. Berger, “An Empirical Analysis of the Costs of Clone- and Platform-Oriented Software Reuse,” in *Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*. ACM, 2020.

[64] J. Krüger, C. Lausberger, I. von Nostitz-Wallwitz, G. Saake, and T. Leich, “Search. Review. Repeat? An Empirical Study of Threats to Replicating SLR Searches,” *Empirical Software Engineering*, vol. 25, no. 1, 2020.

[65] J. Krüger, W. Mahmood, and T. Berger, “Promote-pl: A round-trip engineering process model for adopting and evolving product lines,” in *International Systems and Software Product Line Conference (SPLC)*. ACM, 2020.

[66] R. N. Langlois and P. L. Robertson, "Networks and innovation in a modular system: Lessons from the microcomputer and stereo component industries," *Research Policy*, vol. 21, no. 4, pp. 297–313, 1992.

[67] P. G. Larsen, J. Fitzgerald, J. Woodcock, P. Fritzson, J. Brauer, C. Kleijn, T. Lecomte, M. Pfeil, O. Green, S. Basagiannis, and A. Sadovykh, "Integrated tool chain for model-based design of cyber-physical systems: The into-cps project," in *International Workshop on Modelling, Analysis, and Control of Complex CPS (CPS Data)*. IEEE, 2016.

[68] E. A. Lee, "Cyber physical systems: Design challenges," in *International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*. IEEE, 2008.

[69] M. Li, L. Guan, C. Dickerson, and A. Grigg, "Model-based systems product line engineering with physical design variability for aircraft systems," in *System of Systems Engineering Conference (SoSE)*. IEEE, 2016.

[70] K. Lind and R. Heldal, "Automotive system development using reference architectures," in *IEEE Software Engineering Workshop*, 2012, pp. 42–51.

[71] A. W. Malik, A. U. Rahman, A. Ahmad, and M. M. D. Santos, "Over-the-air software-defined vehicle updates using federated fog environment," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, 2022.

[72] L. Marchezan, E. Rodrigues, W. K. G. Assunção, M. Bernardino, F. P. Basso, and J. Carbonell, "Software product line scoping: A systematic literature review," *Journal of Systems and Software*, vol. 186, 2022.

[73] A. Messac, M. Martinez, and T. Simpson, "Introduction of a product family penalty function using physical programming," *Journal of Mechanical Design*, vol. 124, 2002.

[74] A. Metzger and K. Pohl, "Software product line engineering and variability management: Achievements and challenges," in *Future of Software Engineering (FOSE)*. ACM, 2014.

[75] M. Meyer and A. Lehnerd, "The power of product platforms: Building value and cost leadership," *Journal of Product Innovation Management*, vol. 14, no. 6, 1997.

[76] J. Mikkola and O. Gassmann, "Managing modularity of product architectures," *Engineering Management, IEEE Transactions on*, vol. 50, pp. 204 – 218, 06 2003.

[77] M. Muffatto, "Introducing a platform strategy in product development," *International Journal of Production Economics*, pp. 145–153, 1999.

[78] M. Muffatto and M. Roveda, "Developing product platforms," *Technovation*, vol. 20, no. 11, 2000.

[79] P. Oreizy, N. Medvidovic, and R. N. Taylor, "Architecture-based runtime software evolution," in *International Conference on Software Engineering*, ser. ICSE '98. USA: IEEE Computer Society, 1998, p. 177–186.

[80] S. Otten, T. Glock, C. P. Hohl, and E. Sax, "Model-based variant management in automotive systems engineering," in *International Symposium on Systems Engineering (ISSE)*. IEEE, 2019.

[81] P. Pelliccione, E. Knauss, R. Heldal, M. Ågren, P. Mallozzi, A. Alminger, and D. Borgentun, "A proposal for an automotive architecture framework for volvo cars," in *Workshop on Automotive Systems/Software Architectures (WASA)*. IEEE, 2016.

[82] D. E. Perry and A. L. Wolf, "Foundations for the study of software architecture," *SIGSOFT Softw. Eng. Notes*, vol. 17, no. 4, p. 40–52, 1992.

[83] T. Placho, C. Schmittner, A. Bonitz, and O. Wana, "Management of automotive software updates," *Microprocessors and Microsystems*, vol. 78, 2020.

[84] D. P. Plakhotnikov and E. E. Kotova, "Design and analysis of cyber-physical systems," in *Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*. IEEE, 2021.

[85] K. Pohl, G. Böckle, and F. Van Der Linden, *Software Product Line Engineering*. Springer, 2005.

[86] A. Poth, "Product line requirements engineering in the context of process aspects in organizations with various domains," *Software Process: Improvement and Practice*, vol. 14, no. 6, 2009.

[87] B. Poudel and A. Munir, "Design and evaluation of a reconfigurable ecu architecture for secure and dependable automotive cps," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, 2021.

[88] P. Queiroz and R. T. Braga, "A critical embedded system product line model-based approach," in *International Conference on Software Engineering and Knowledge Engineering (SEKE)*. Knowledge Systems Institute Graduate School, 2014.

[89] R. Rabiser and A. Zoitl, "Towards mastering variability in software-intensive cyber-physical production systems," *Procedia Computer Science*, vol. 180, 2021.

[90] U. Raubold, *Lebenszyklusmanagement in der Automobilindustrie*. Springer, 2011.

[91] S. Raue, "Systemorientierung in der modellbasierten modularen e/e-architekturentwicklung," Ph.D. dissertation, Eberhard Karl University Tübingen, 2019.

[92] D. Robertson and K. Ulrich, "Planning for product platforms," *Sloan Management Review*, vol. 39, no. 4, 1998.

[93] J. Rubin, K. Czarnecki, and M. Chechik, "Managing Cloned Variants: A Framework and Experience," in *International Software Product Line Conference (SPLC)*. ACM, 2013.

[94] R. Sanchez, "Strategic product creation: Managing new interactions of technology, markets, and organizations," *European Management Journal*, vol. 14, no. 2, pp. 121–138, 1996.

[95] R. G. Sanfelice, "Analysis and design of cyber-physical systems: A hybrid control systems approach," in *Cyber-Physical Systems: From Theory to Practice*. CRC Press, 2015.

[96] A. Sangiovanni-Vincentelli and G. Martin, "Platform-based design and software design methodology for embedded systems," *IEEE Design & Test of Computers*, vol. 18, no. 6, pp. 23–33, 2001.

[97] K. Schmid and I. John, "A customizable approach to full lifecycle variability management," *Science of Computer Programming*, vol. 53, no. 3, pp. 259–284, 2004, software Variability Management.

[98] K. Schmid and M. Verlage, "The Economic Impact of Product Line Adoption and Evolution," *IEEE Software*, vol. 19, no. 4, 2002.

[99] G. Schuh and M. Riesener, *Produktkomplexität managen*. Hanser, 2017.

[100] M. Shaw, R. DeLine, D. Klein, T. Ross, D. Young, and G. Zelesnik, "Abstractions for software architecture and tools to support them," *IEEE Transactions on Software Engineering*, vol. 21, no. 4, pp. 314–335, 1995.

[101] T. W. Simpson, "Product platform design and customization: Status and promise," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 18, no. 1, 2004.

[102] T. W. Simpson, J. R. A. Maier, and F. Mistree, "Product platform design: method and application," *Research in Engineering Design*, vol. 13, pp. 2–22, 2001.

[103] L. Sion, D. Van Landuyt, W. Joosen, and G. de Jong, "Systematic quality trade-off support in the software product-line configuration process," in *International Software Product Line Conference (SPLC)*. ACM, 2016.

[104] Ş. Stănciulescu, S. Schulze, and A. Wąsowski, "Forked and Integrated Variants in an Open-Source Firmware Project," in *International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2015.

[105] J. Stark, *Product Lifecycle Management*. Springer, 2020.

[106] S. Thiel, M. A. Babar, G. Botterweck, and L. O'Brien, "Software product lines in automotive systems engineering," *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, vol. 1, no. 1, 2009.

[107] T. Thüm, S. Apel, C. Kästner, I. Schaefer, and G. Saake, "A classification and survey of analysis strategies for software product lines," *ACM Computing Surveys*, vol. 47, pp. 1–45, 07 2014.

[108] K. Ulrich, "The role of product architecture in the manufacturing firm," *Research Policy*, vol. 24, no. 3, pp. 419–440, 1995.

[109] S. ur Rehman, A. Iannella, and V. Gruhn, "A security based reference architecture for cyber-physical systems," in *Applied Informatics*. Springer, 2018.

[110] F. Van der Linden, K. Schmid, and E. Rommes, *Software Product Lines in Action: the Best Industrial Practice in Product Line Engineering*. Springer, 2007.

[111] J. van Gurp, J. Bosch, and M. Svahnberg, "On the notion of variability in software product lines," in *Working IEEE/IFIP Conference on Software Architecture*, 2001, pp. 45–54.

[112] T. Vietor and C. Stechert, *Produktarten zur Rationalisierung des Entwicklungs- und Konstruktionsprozesses*. Springer, 2013.

[113] M. Voelter and I. Groher, "Product line implementation using aspect-oriented and model-driven software development," in *International Systems and Software Product Line Conference (SPLC)*. USA: IEEE, 2007, p. 233–242.

[114] P. Wallin, S. Johnsson, and J. Axelsson, "Issues related to development of e/e product line architectures in heavy vehicles," in *Hawaii International Conference on System Sciences (HICSS)*. IEEE, 2009.

[115] T. R. Wanasinghe, M. Galagedarage Don, R. Arunthavanathan, and R. G. Gosine, "Industry 4.0 based process data analytics platform," in *Methods to Assess and Manage Process Safety in Digitalized Process System*. Elsevier, 2022.

[116] B. Wilhelm, "Platform and modular concepts at volkswagen — their effects on the assembly process." Springer, 1997, pp. 146–156.

[117] C. Wohlin, "Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering," in *International Conference on Evaluation and Assessment in Software Engineering (EASE)*. ACM, 2014.

[118] P. Zellmer, L. Holsten, T. Leich, and J. Krüger, "Product-structuring concepts for automotive platforms: A systematic mapping study," in *International Systems and Software Product Line Conference (SPLC)*. ACM, 2023, p. 170–181.

[119] P. Zellmer, L. Holsten, R. May, and T. Leich, "A practitioners perspective on addressing cyber security and variability challenges in modern automotive systems," in *International Working Conference on Variability Modelling of Software-Intensive Systems*, ser. VaMoS '24. ACM, 2024, p. 129–133.

[120] P. Zellmer, J. Krüger, and T. Leich, "Decision making for managing automotive platforms: An interview survey on the state-of-practice," in *International Conference on the Foundations of Software Engineering*, ser. FSE 2024. ACM, 2024, p. 318–328.

[121] T. Şahin, T. Huth, J. Axmann, and T. Vietor, "A methodology for value-oriented strategic release planning to provide continuous product upgrading," in *International Conference on Industrial Engineering and Engineering Management (IEEM)*. IEEE, 2020.

Philipp Zellmer was born in Magdeburg, Germany in 1994. He received the M.S. degree in Business Administration and Engineering from the Faculty of Economics and Business Administration, Chemnitz University of Technology, Germany in 2020. He participated in an industrial doctoral program at Volkswagen AG, which is conducted in partnership with the Doctoral Center for Engineering Sciences and Information Technologies at Harz University of Applied Sciences. His research primarily focuses on supporting the practical applicability of concepts and methods, especially in the context of electrics/electronics platforms, for managing variant rich vehicle portfolios throughout their life cycle.

Lennart Holsten was born in Hannover, Germany in 1996. He received the M.S. degree in industrial engineering from the Faculty of Mechanical Engineering, Technical University Braunschweig, Germany in 2021. He is currently participating in an industrial doctoral program at Volkswagen AG, which is conducted in partnership with the Doctoral Center for Engineering Sciences and Information Technologies at Harz University of Applied Sciences. His research primarily focuses on optimizing the management of variant rich vehicle portfolios through the utilization of electrics/electronics platforms and their application in practical contexts.

Jacob Krüger is Assistant Professor for software engineering at Eindhoven University of Technology, The Netherlands. He previously worked at Harz University of Applied Sciences Wernigerode, Germany, Otto-von-Guericke University Magdeburg, Germany, and Ruhr-University Bochum, Germany—and visited Chalmers University of Technology | University of Gothenburg, Sweden, as well as the University of Toronto, Canada. He specializes in the development and evolution of (variant-rich) software systems, aiming to research the interplay of human cognition and software quality in this context.

Thomas Leich received his diploma in Business Information Systems and his PhD from the University of Magdeburg in 2002 and 2012, respectively. Since 2013 he is general manager of the METOP GmbH. Since 2014 Thomas Leich is professor at the chair of Business Information Systems at Harz University of Applied Sciences. His research interests include understanding of systems with variability, including work on implementation mechanisms, tools, empirical evaluations, and refactoring as well as measurement of program comprehension.