Systematic Review

Managed Evolution of Automotive Software Product Line Architectures: A Systematic Literature Study

Christoph Knieke [1,*], Andreas Rausch [1], Mirco Schindler [1], Arthur Strasser [1] and Martin Vogel [1]

[1] Clausthal University of Technology, Institute for Software and Systems Engineering, Clausthal-Zellerfeld, Germany
* Correspondence: christoph.knieke@tu-clausthal.de (C.K.)

Abstract: The rapidly growing number of software-based features in the automotive domain as well as the special requirements in this domain ask for dedicated engineering approaches, models, and processes. Nowadays, software development in the automotive sector is generally developed as product line development, in which major parts of the software are kept adaptable in order to enable reusability of the software in different vehicle variants. In addition, reuse also plays an important role in the development of new vehicle generations in order to reduce development costs. Today, a high number of methods and techniques exist to support the product line driven development of software in the automotive sector. However, these approaches generally consider only partial aspects of development. In this paper we present an in-depth literature study based on a conceptual model of artifacts and activities for the managed evolution of automotive software product line architectures. We are interested into the coverage of the particular aspects of the conceptual model and, thus, the fields covered in current research and research gaps, respectively. Furthermore, we aim to identify the methods and techniques used to implement automotive software product lines in general, and their usage scope in particular. As a result, the in-depth review reveals that none of the studies represents a holistic approach for managed evolution of automotive software product lines. In addition, approaches from agile software development are of growing interest in this field.

Keywords: Automotive Software Engineering; Software Product Lines; Systematic Literature Study

Citation: Knieke, C.; Rausch, A.; Schindler, M.; Strasser, A; Vogel, M. Managed Evolution of Automotive Software Product Line Architectures: A Systematic Literature Study. *Preprints* **2022**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

1. Introduction

The automobile has become the technically most complex consumer product [1]. The fulfillment of increasing customer requirements and strict legal requirements with regard to the reduction of fuel consumption and pollutant emissions, as well as the higher demands on safety and new driver assistance systems resulted in a steady increase in the deployment of onboard electronics systems and software. Software-intensive systems and functions are the major drivers for innovations in cars today [2]. In premium vehicles software is responsible for up to 80% of the innovation [3]. Electronics and software account for up to 40% of the production costs of a car [4].

The requirements for automotive electronics differ significantly from other areas of consumer electronics (cf., e.g., [5], [6]). For automotive functions hard real-time requirements exist while the storage and computational power on ECUs has to be kept as small as possible. Systems in an automobile must be reliable, safe and secure in all situations as system failures can endanger human life. In addition, long product life cycles exist: the OEM has the duty to offer service and spare parts for at least 15 years after the purchase of a vehicle due to the long product life cycles. In contrast, software is changed at comparatively short intervals. During the production period and even during the development phase, many new versions of a piece of software are developed. As a consequence of short innovation and long life cycles a huge number of versions and configurations exist, which inter alia makes maintenance very difficult.

In order to reduce software development costs, the automotive industry aims for a high degree of software reuse. Reuse is achieved both through cross-product development for different vehicle variants and through reuse in subsequent products. However, the

increasing complexity and variability of automotive software systems is making it increasingly difficult to reuse and extend these systems. From one vehicle generation to the next, functionality differs mostly not more than 10%, while much more than 10% of the software is rewritten [6]. This has led to a growing interest in software product line approaches to automotive systems [7].

Since the 1990s software product lines have been introduced as a major addition to existing reuse approaches [8–11]. Clements et al. [11] define a software product line (SPL) as a family of systems that share a common set of core technical assets, with preplanned extensions and variations to address the needs of specific customers or market segments.

In general, software product line engineering consists of two key processes, domain engineering and application engineering [12]. The aim of the domain engineering process is to define and realize the commonality and the variability of the software product line. The process of application engineering is responsible for deriving product line applications from the platform established in domain engineering by exploiting the variability of the software product line [12].

Nowadays, many challenges in the area of automotive software product line development are mentioned in literature [7,13] (see Section 2.1.1). Motivated by the huge set of challenges in this domain, many studies exist proposing particular methods and techniques to support the product line driven development of software in the automotive sector (see, e.g., [13]). However, these approaches generally consider only partial aspects of development. An overview on the set of available methods and techniques would help researchers and practitioners in the field of automotive software product line engineering. As Raatikainen et al. [14] state in their study, there is a need for a combination of existing solutions in the area of software product lines rather than for novel approaches. This study examines existing approaches with respect to a holistic model, so that one gets an order along a process view (which is domain specific). For the introduction of a software product line approach it is crucial to know which activities (process steps) are supported by it and which are not.

Moreover, existing studies either focus on the domain of automotive software engineering in general, or they address the whole field of software product line engineering without regarding the specific requirements of the automotive domain. Thus, a study providing a holistic view on software product line engineering in the automotive domain is still missing.

1.1. Research Approach and Contribution

In this paper we present an in-depth literature review based on a conceptual model of artifacts and activities for the managed evolution of automotive software product line architectures.

Our study is grounded in [13]. For this, following Kitchenham et al. [15] and Petersen et al. [16,17], we use [13] as a so-called *scoping study* that we utilize to investigate a specific topic in more detail. Since the study in [13] only considers papers until 2015, we have additionally included relevant papers from 2016-2021 in our study.

We aim at collecting information about current state-of-the-art in holistic approaches for managed evolution of automotive software product lines and use our conceptual model as a reference model to evaluate the state-of-the-art. We are interested into the coverage of the particular aspects of the conceptual model and, thus, the fields covered in current research and research gaps, respectively. Furthermore, we aim to identify the methods and techniques used to implement automotive software product lines in general, and their usage scope in particular. The paper supports engineers to get an overview on existing methods and techniques in automotive software product line engineering and helps researchers to find research gaps.

1.2. Outline

The paper is organized as follows: Section 2 provides a conceptual model of artifacts and activities on automotive software product line engineering and summarizes the related work. In Section 3, we describe our research design, and present the results of our study in Section 4. We conclude the paper in Section 5.

2. Related Work & Background

2.1. Related Work

In this section, we give an overview on the related work. We regard existing surveys on automotive software engineering and software product lines in general. Subsequently, we define a conceptual model on managed evolution of software product line architectures as a basis for our systematic literature study.

2.1.1. Automotive Software Engineering

Haghighatkhaha et al. [13] published a systematic mapping study that analyzes and classifies the literature related to the field of automotive software engineering. This review includes 679 articles from multiple research sub-areas, published between 1990 and 2015. They analyze research activities, topics, types, and methods and reveal research gaps in automotive software engineering. A classification of the 679 articles is listed in an excel sheet including general information for each article like abstract, title, year, authors, and research topic.

Pretschner et al. [6] provided a comprehensive overview on the state of the art in automotive software engineering. They identified research challenges in automotive software engineering, in particular the integration, evolution, maintenance, and reuse, and explored potential benefits of a seamless model-based development process as a possible solution. Furthermore, the study provides a roadmap for research in this area.

Clarke et al. [18] identified important areas in software engineering that will have a significant impact on future automotive systems. The authors introduced global software development, software product lines, service-oriented architectures, and mathematical approaches applied to software engineering as possible future research directions for the automotive domain.

Grimm [19] discussed major challenges of automotive software engineering and the most important technological core competencies required to meet these challenges. Amongst other things, future work will focus on the following fields: Elaboration of a software product line approach for future in-vehicle software architectures, model-based development of distributed systems, and integration of processes, methods and tools from the different areas of mechanical, electrical, and software engineering.

Gruszczynski [20] gave an overview on software engineering technologies in the automotive industry and identified future research directions. Fabbrini et al. [21] provided a picture of the achievements and the open issues in the European automotive industry and suggested future research directions.

Thiel et al. [7] presented some challenges that automotive engineering faces today and discuss contributions software product line approaches could make to provide solutions for these challenges.

Antinyan, in his paper “Revealing the Complexity of Automotive Software” [22] identified four aspects that an approach to automotive software evolution must mitigate. These are the complexity of the requirements, the complexity of source code, the architecture complexity, and the complexity in creating variants. All these areas are represented by the conceptual model (see Section 2.2), which serves as the baseline of this study.

2.1.2. Software Product Line Engineering

Since the 1990s software product lines have been introduced as a major addition to existing reuse approaches [8] [9] [10] [11]. Clements et al. [11] define a software product line (SPL) as a family of systems that share a common set of core technical assets, with

preplanned extensions and variations to address the needs of specific customers or market segments.

Pohl et al. [12] propose a holistic approach on software product line engineering consisting of two key processes, domain engineering and application engineering. Domain engineering process aims at defining and realizing the variability and commonality of the software product line. The process of application engineering aims at deriving product line applications from the platform established in domain engineering by exploiting the variability of the software product line [12].

An important activity in software product line engineering constitutes variability management [11,12]. Most existing approaches in variability management can be classified as feature modeling and decision modeling [23]. The main difference between both approaches is that feature modeling supports both commonality and variability modeling, whereas decision modeling focuses exclusively on variability modeling [24].

In the following, we present surveys on various topics of SPL:

Harman et al. [25] presented a survey on Search Based Software Engineering (SBSE) for SPLs and highlighted some directions for future work. They identified the most active areas in SBSE for SPLs: SPL testing, SPL feature selection, product line architecture (PLA) improvement, and SPL feature extraction.

Furthermore, several surveys on product-line testing have been conducted: Engström and Runeson [26] presented a systematic mapping study on SPL testing. The main challenges identified in the paper are the large number of tests, the balance between effort for reusable components and concrete products, and handling variability. Lee et al. [27] surveyed the current SPL testing approaches and highlight the challenges and key research perspectives in SPL testing. They defined a reference SPL testing processes as survey framework. Oster et al. [28] also addressed SPL testing and presented a survey on the state-of-the-art of model-based testing (MBT) approaches for SPL. They defined a conceptual process model for SPL testing which is used for a comparison of the different testing approaches. Furthermore, they highlighted the challenges and open research topics in SPL testing.

Thüm et al. [29] proposed a classification of SPL analyses and survey and classify 123 existing approaches for the analysis of SPL. They also provided a research agenda to guide future research on SPL analyses.

Chen et al. [30] discussed the findings from a systematic literature review in variability management in SPL. They presented the chronological backgrounds of various approaches in variability management and identified certain gaps that need to be filled by future research: Amongst other things, they concluded, that only a few approaches address systematic process support for variability management and that there is only limited support for evolution of variability. In addition, they stated the inability of most approaches to scale to large and complex product lines.

Schobbens et al. [31] presented a survey on existing feature diagram variants. Based on the regarded feature diagram variants, they proposed a generic formalization of the syntax and semantics of feature diagrams.

Also recent systematic literature studies [32], [33] and [14] are still identifying research gaps. Chacón-Luna et al. [32] criticize the accuracy of empirical studies in particular and the fact that most of the approaches are evaluated on artificial scenarios. Marques et al. [33] also note that while case studies are prevalent, only a few industry studies of an appropriate size are publicly available. Furthermore, SPL approaches cannot be compared due to a lack of consensus regarding their formalization. Raatikainen et al. [14] suggest that the goal must be not to develop new and novel SPL methods in the future but to make better use of the existing ones in terms of actionability, context-sensitivity, and evidence and quality in the resulting syntheses.

2.2. Conceptual Model for Managed Evolution of Automotive Software Product Line Architectures

As discussed in [34] classical holistic approaches on software product line engineering, like [12], have to be adapted to the special requirements of the automotive domain: In the automotive sector, it is not possible to carry out all further developments within the product line. Rather, there may be further developments that do not take place in the product line but at the level of the individual products. The reasons for this may be the high time and cost pressure, but also the fact that sophisticated further developments are initially to be tested within the scope of a prototype implementation. These further developments, which are separate from the product line, have to be transferred into product line development at a later stage. This goes beyond the classical domain/application engineering approaches, like [12].

In addition, we are often facing an eroded software architecture. Thus, we first have to repair the architecture as discussed by Cool et al. [35]. Architecture repair typically involves the two approaches *recovery* and *discovery*. Recovery is based on reverse engineering techniques by which the implemented architecture is extracted from source artifacts whereas discovery hypothesizes its intended architecture [36].

A challenge is to minimize architecture erosion in the long term: The product line architecture is designed initially and develops over time. Further development must ensure that the product architecture remains consistent with the product line architecture. Thus, in order to prevent architecture erosion in the future, architecture conformance checking is required for all further developments.

Based on these prerequisites we use a conceptual model with activities for managed evolution of automotive software product line architectures (cf. [35] and [34]) as shown in Figure 1. Figure 1 depicts in the left part the activity of discovery and recovery. This activity aims at repairing an initially eroded software architecture. It is performed once before the long-term development cycle can begin. The long term evolution cycle consists of two layers: The cycle `Product line (PL)` (see Figure 1) contains the activities for the development of the product line. Cycle `Product (P)` includes the activities for the development on the product specific level. These two cycles enable a parallel development, and even the activities within one cycle can be performed independently. The dependencies of an activity on artifacts of the previous activity are indicated by the circular arrow in the middle of the two cycles. Despite this, a parallel execution of single activities is also possible: A new PLA can be developed in activity `PL-Design` while an implementation is realized in activity `PL-Implement` An external decision-making process is necessary to start a new prototyping or to transfer a prototype implementation back to the product line. Therefore, we have introduced the large, blue arrows between the two layers.

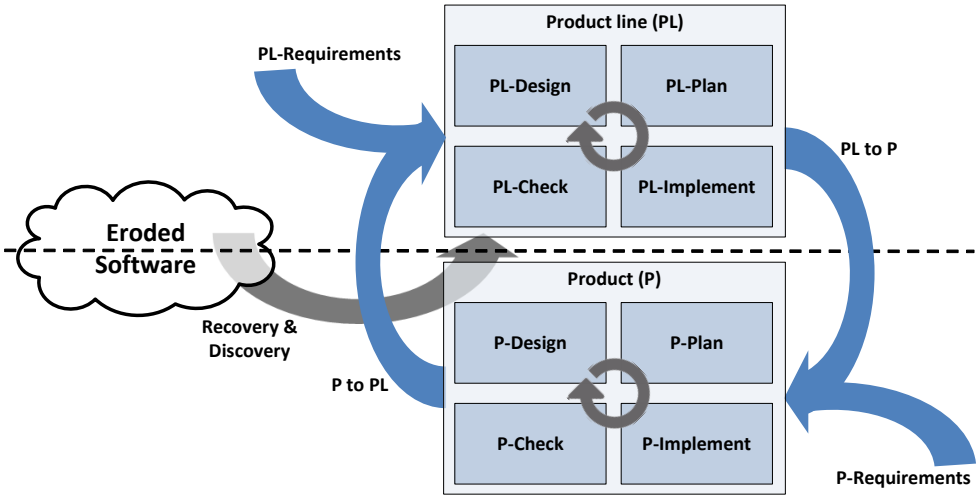

Figure 1. Conceptual model for managed evolution

Next, we will briefly introduce the activities of the conceptual model in Figure 1. Table 1 (cf. [35] and [34]) summarizes the objectives of the 13 activities, including input and output artifacts.

Table 1: Explanation of the activities in Figure 1.

Activity	Objective	Input/Output
Recovery & Discovery	Recovery of the implemented PLA from the source artifacts (developed products) and discovery of the intended PLA.	*Input:* Source artifacts (developed products). *Output:* Implemented and intended PLA.
PL-Requirements	Specification and validation of software system and software component requirements by requirements engineering.	*Input:* Requirements. *Output:* Software system and software component requirements.
PL-Design	Further development of PLA with consideration of design principles. Application of measuring techniques to assess quality of PLA.	*Input:* Software system / component requirements and documentation from product development. *Output:* New PLA (called "PLA vision").
PL-Plan	Planning of a set of iterations of further development toward the PLA vision taking all affected projects into account.	*Input:* PLA vision. *Output:* Development plan including the planned order of module implementations and the planned related projects.
PL-Implement	Implementation including testing as specified by the development plan for product line development.	*Input:* Development plan for product line. *Output:* Implemented module versions.
PL-Check	Minimization of product architecture erosion by architecture conformance checking based on architecture rules.	*Input:* Architecture rules and set of implemented modules to be checked. *Output:* Check results.
PL to P	Defining a project plan by selecting a project from the the product line.	*Input:* Development plan for product line. *Output:* Project plan.
P-Requirements	Specification of special requirements for a certain vehicle product including vehicle related parameter settings.	*Input:* Requirements in particular from calibration engineers. *Output:* Vehicle related requirements.
P-Design	Designing product architecture and performing architecture adaptations taking product specific requirements into account. Compliance checking with PLA to minimize erosion.	*Input:* Project plan and product specific requirements. *Output:* Planned product architecture.
P-Plan	Definition of iterations to be performed on product level toward the planned product architecture.	*Input:* Product architecture. *Output:* Development plan for product development.
P-Implement	Product specific implementations including testing as specified by the development plan for product development.	*Input:* Development plan for product development. *Output:* Implemented module versions.
P-Check	Architecture conformance checking between PLA and PA.	*Input:* Architecture rules and set of implemented modules to be checked. *Output:* Check results.
P to PL	Providing product related information of developed product for integration into product line development.	*Input:* Developed product. *Output:* Product documentation and implementation artifacts of developed products.

The cycle for the product line architecture needs as input the requirements from requirements engineering (`PL-Requirements`) and all the artifacts developed on product level. Activities `PL-Plan` and `PL-Design` serve to design, plan and evolve the PLA.

In `PL-Implement` the planned implementation artifacts are implemented on level of the product line, whereas in `P-Implement` the implementation of the product specific artifacts takes place. To create a fully functional software status for a particular vehicle

project, the project plan is transferred (`PL to P`), which contains module descriptions and descriptions of the integration plan of the logical product architecture with the associated module versions.

Furthermore, special requirements for a particular project are taken into account (`P-Requirements`). To build a new product, we start with an initially planned PA derived from the product line (`P-Design`). Iterations to be performed are planned in `P-Plan`. An iteration contains the planned implementations and the elements of the PA.

Each planned project refers to a set of implementation artifacts, called modules constituting the product architecture (PA). `P-Check` and `PL-Check` minimize PA erosion by architecture conformance checking. In `P-Design`, we apply architecture conformance checking to check conformance between the planned PA and the PLA.

3. Research Design

In this section, we describe our research design. Section 3.1 describes our overall approach, followed by the research questions in Section 3.2. Section 3.3 describes the data collection procedures. Finally, in Section 3.4, we describe the analysis procedures and discuss the validity procedures implemented.

3.1. Research Method

The study at hand presents an in-depth literature study, which is grounded in [13]. For this, following Kitchenham et al. [15] and Petersen et al. [16,17], we use [13] as a so-called *scoping study* that we utilize to investigate a specific topic in more detail. Since the study in [13] only considers papers unit 2015, we have additionally included relevant papers from 2016-2021 in our study. To investigate our research questions (Section 3.2), we implemented the following procedure:

Step 1 In the first step, we analyzed the scoping study [13] and carried out a data cleaning based on the inclusion/exclusion criteria resulting from the research questions.

Step 2 In the second step, we removed multiple occurrences of papers in the data set following the steps described in [37].

Step 3 On order to include further relevant papers from 2016-2021, we collected papers by defining an applying a search string. Similar to steps 1 and 2, we carried out a data cleaning on the collected papers.

Step 4 In the fourth step, we read the papers and applied the *rigor-relevance* model as proposed by Ivarsson and Gorschek [38].

Step 5 The fifth step comprised a quality assessment following the approach described in [39] (see also App. B).

Step 5 In the sixth step, we prepared the in-depth review. In this regard, we used a refined conceptual model for software product lines [34], which we used to further classify the papers in the data set.

Step 7 Finally, we used the aforementioned conceptual model and the categorized papers to conduct the in-depth review to answer the research questions.

The research approach above was implemented in a team of researchers with well-defined task distribution to improve the validity of the findings.

3.2. Research Questions

With the paper at hand, we aim to understand the current state-of-the-art in holistic approaches for a managed evolution of software product line architecture in automotive software engineering. Specifically, we define our working hypothesis as follows: *There is no holistic approach for a managed evolution of automotive software product lines.*

Table 2: Research questions addressed with the study at hand.

	Research question and rationale
RQ_1	*What is the current state-of-the-art in holistic approaches for managed evolution of automotive software product lines?* We aim at collecting information about such holistic approaches and use our conceptual model (cf. Section 2.2) as a reference model to evaluate the current state-of-the-art. Specifically, we are interested into the coverage of the particular aspects of the conceptual model and, thus, the fields covered in current research and research gaps, respectively.
RQ_2	*What particular methods and techniques are used to implement a managed evolution of automotive software product lines?* We aim to identify the methods and techniques used to implement automotive software product lines in general, and their usage scope in particular. For this, we analyze the available literature and categorize and evaluate the contributions found according to a given schema (Section 2.2).

3.3. Data Collection Procedures

In this section, we describe the data collection procedures. The data collection is based on a previously conducted study [13], which we use as a scoping study. In the following, we provide a brief summary of the scoping study's contribution in Section 3.3.1, before we describe the selection process of papers relevant to the study at hand in Section 3.3.2. In addition, we describe how we included papers from 2016-2021 3.3.3, as the conducted study in [13] only considers papers until 2015. Finally, in Section 3.3.4, we describe how we selected the final set of papers for analysis.

3.3.1. Overview of the Scoping Study

As basic data source, we use the scoping study [13] and the complementing published dataset. The scoping study has collected papers on Automotive Software Engineering, which were categorized into the seven research areas listed in Table 3.

In total, the scoping study [13] includes 679 papers, which were categorized into seven research areas and, finally, were found add rings 14 specific research topics. This dataset served as input for the data collection, which is explained in detail in the following sections.

3.3.2. Study Selection from the Scoping Study

As introduced in Section 3.2, the scope of the study at hand is the field of software product lines. For this, we used the scoping study and applied a multi-staged selection procedure for studies of interest:

1. Analyze the scoping study's data and identify all papers on software product lines (incl. synonyms):
   (a) Selection of papers based on title and abstract
   (b) Selection based on keywords[1] (incl. variants, upper-/lower case, etc.): SPL, family, reference architecture, variability, variant model, variability management, feature model, feature tree, feature-oriented, derivate
   (c) *Result:* This stage resulted into 87 paper (candidates); most of the papers selected are in the REU group (Table 3)
2. Rating of the study candidates (Section 3.3.2)
   (a) Application of the rigor-relevance model [38]
   (b) Definition of a Threshold and selection of papers
   (c) Final data cleaning and preparation of the in-depth analysis

[1] While testing the selection criteria, we decided to exclude the keywords *reusability*, *reuse*, *derive*, and *feature*, since they generated too many false positives.

Table 3: Automotive Software Engineering research areas according to ISO/IEC 12207 PRM as found in [13].

Research area and topics	**Studies per area**
Agreement Processes (AGR)	5
• Agreement Support Group	
Organizational Project-Enabling Processes (ORG)	48
• Organizational Project-Enabling Support Group	
Project Processes (PRO)	14
• Project Support Group	
Technical and Software Implementation Processes[a] (ENG/DEV)	439
• System/Software Architecture and Design (131 studies).	
• System/Software Qualification Testing (127 studies)	
• Software Implementation (62 studies).	
• System/Software Integration (44 studies)	
• System/Software Requirement Engineering (35 studies).	
• Software Construction (22 studies).	
• Software Maintenance (18 studies).	
Software Support Processes (SUP)	122
• Software Verification and Validation (71 studies).	
• Software Quality Assurance and Review (48 studies).	
• Software Documentation and Configuration Management (3 studies).	
Software Reuse Processes (REU)	72
• Software Reuse	

[a] SUMMARIZED

3. In-depth analysis (Section 3.4)

3.3.3. Collection and selection of papers from 2016-2021

To include papers in this work that were published after 2015, we performed another search for relevant papers. For this purpose, we defined a suitable search string and applied it to Google Scholar. In particular, we performed the following steps:

1. Definition of a suitable search string.
2. *Result:* `"automotive software" AND "software product line"`.
3. Application of the search string to Google Scholar for the years 2016-2021.
4. *Result:* This stage resulted in 263 papers (candidates).
5. Selection of papers based on title and abstract, done by two reviewers.
6. *Result:* This stage resulted in 20 papers (candidates).

To distinguish the papers from the scoping study [13] and the selected papers from 2016-2021, we gave the prefix P to the ID of the scoping study's papers and the prefix E to the papers of the extended search.

3.3.4. Final Paper Selection

In the final paper selection, we analyzed the 107 selected candidate studies and, in a first step, applied the rigor-relevance model as proposed by Invarsson and Gorschek [38]. This model grades papers on the two parameters *rigor* (FROM-TO) and *relevance*

(FROM-TO). To select the studies of interest, we defined the threshold $rigor + relevance \geq 2$ to ensure that:

1. All high-quality papers (i.e., high-scored papers) are in the result set
2. Papers that have a high relevance score but a poor rigor score are included
3. Medium-scored but balanced papers are included

Applying the rigor relevance model to the candidate studies, we selected 56 studies. These studies were (again) checked with a particular focus on multiple occurrences. In order to finally clean the dataset, we decided to prefer journal papers to conference papers, as we assume follow-up special issue papers to have a higher maturity/quality. Eventually, we selected *51* primary studies for inclusion into the data analysis. The final evaluation and classification of the different studies is summarized in Table 4.

3.4. Analysis and Validity Procedures

The analysis procedures applied to the final dataset of 51 papers also followed a multi-staged approach. In our analysis procedure, we applied a number standard analyses on the data already provided by the original scoping study [13]. In particular, we used the provided data for analyzing the *research type facets* [78] and the *contribution type facets* [16,17]. Furthermore, we applied the rigor-relevance model [38] for implementing study selection (Section 3.3.4) and further data analyses.

Grounded in these basic measures, we used the conceptual model for a managed evolution from Section 2.2 as a classification schema. Five researchers classified the papers according to the conceptual model from Section 2.2, which we used as a classification schema. For each element in the classification schema, the individual researcher had to state whether or not a study makes a major contribution to a specific element, e.g., *product line design* or *product implementation*. The results were integrated and checked for agreement using Fleiss' κ [92]; the overall agreement in the studies' classification was 0.72 (substantial agreement). This classification resulted into three categories:

Category 1 This category includes those papers for which all five researchers agreed that a study has a major contribution in a specific category. Finally, 8 papers were assigned to this category.

Category 2 This category includes those papers for which three or four researchers agreed that a study has a major contribution in a specific category. Finally, 30 papers were assigned to this category.

Category 3 This category includes those papers for which one or two researchers only agreed that a study has a major contribution in a specific category. Finally, 13 papers were assigned to this category.

The categories above are used in further in-depth analyses to, notably, evaluate the specific contributions and the maturity of the contributions of the individual studies. Finally, we used the classification schema from Section 2.2 and the researchers' rating to generate a mapping table, which guides the in-depth content analysis.

As we reused an already published dataset resulting from an independently conducted scoping study [13], we implemented several measures to improve the validity of our findings. First and foremost, we relied on researcher triangulation, i.e., we always ensured that one researcher from the team was not involved in a task and performed the quality assurance of that very task, e.g., in the rating of the contributions of the study (Section 3.4), five researchers evaluated the studies and a sixth researcher checked the evaluations and computed the agreement levels. Furthermore, we shuffled the teams, e.g., data collection, quality assessment, and data analysis were performed by different teams of two to three researchers.

4. Study Results & Discussion

This section summarizes the results of our analysis. First, we provide general demographic information about the dataset, before we answer the individual research questions.

Table 4: Evaluation and classification of the primary studies selected for analysis.

Id	Ref	RTF[a]	CTF[b]	Rigor	Relevance
P8	[40]	Evaluation	Model	1.0	2.0
P22	[41]	Solution Proposal	Framework/Method/Technique	1.5	2.0
P43	[42]	Evaluation	Lessons Learned	3.0	3.0
P71	[43]	Evaluation	Framework/Method/Technique	1.5	2.0
P72	[44]	Evaluation	Lessons Learned	3.0	3.0
P79	[45]	Evaluation	Framework/Method/Technique	1.0	2.0
P86	[46]	Evaluation	Lessons Learned	2.0	4.0
P90	[47]	Evaluation	Model	2.0	3.0
P94	[48]	Evaluation	Framework/Method/Technique	1.0	2.0
P100	[49]	Validation	Framework/Method/Technique	2.5	1.0
P129	[50]	Evaluation	Model	2.0	4.0
P173	[51]	Evaluation	Framework/Method/Technique	1.0	2.0
P215	[52]	Evaluation	Lessons Learned	2.5	4.0
P218	[53]	Experience Paper	Lessons Learned	0.5	2.0
P220	[54]	Experience Paper	Lessons Learned	0.5	2.0
P221	[55]	Experience Paper	Lessons Learned	0.5	2.0
P223	[56]	Experience Paper	Framework/Method/Technique	1.0	2.0
P279	[57]	Evaluation	Framework/Method/Technique	1.0	3.0
P281	[58]	Validation	Guideline	1.0	2.0
P285	[59]	Evaluation	Framework/Method/Technique	1.5	4.0
P289	[60]	Evaluation	Framework/Method/Technique	1.0	2.0
P300	[61]	Evaluation	Lessons Learned	1.0	2.0
P310	[62]	Evaluation	Framework/Method/Technique	1.0	2.0
P332	[63]	Evaluation	Framework/Method/Technique	1.0	1.0
P343	[64]	Evaluation	Framework/Method/Technique	1.0	2.0
P365	[65]	Evaluation	Framework/Method/Technique	1.0	1.0
P377	[66]	Solution Proposal	Framework/Method/Technique	1.0	1.0
P404	[67]	Evaluation	Lessons Learned	0.5	2.0
P468	[68]	Experience Paper	Lessons Learned	1.0	3.0
P493	[69]	Experience Paper	Framework/Method/Technique	2.0	3.0
P503	[70]	Experience Paper	Lessons Learned	0.0	2.0
P580	[71]	Solution Proposal	Framework/Method/Technique	1.0	3.0
P588	[72]	Evaluation	Framework/Method/Technique	1.0	3.0
P589	[73]	Evaluation	Framework/Method/Technique	1.0	2.0
P660	[74]	Evaluation	Framework/Method/Technique	1.0	2.0
P580	[71]	Solution Proposal	Framework/Method/Technique	1.0	3.0
P588	[72]	Evaluation	Framework/Method/Technique	1.0	3.0
P589	[73]	Evaluation	Framework/Method/Technique	1.0	2.0
P660	[74]	Evaluation	Framework/Method/Technique	1.0	2.0
E8	[75]	Evaluation	Framework/Method/Technique	1.0	2.0
E53	[76]	Evaluation	Framework/Method/Technique	1.0	3.0
E59	[77]	Evaluation	Lessons Learned	2.0	4.0

[a] Research type facet, according to [78]
[b] Contribution type facet, according to [16,17]

Id	Ref	RTF[a]	CTF[b]	Rigor	Relevance
E66	[79]	Evaluation	Framework/Method/Technique	2.0	2.0
E127	[80]	Evaluation	Framework/Method/Technique	1.0	4.0
E130	[81]	Evaluation	Framework/Method/Technique	1.0	4.0
E132	[82]	Evaluation	Framework/Method/Technique	2.0	2.0
E173	[83]	Experience Paper	Guideline	3.0	2.0
E175	[84]	Experience Paper	Lessons Learned	2.5	2.0
E177	[85]	Experience Paper	Lessons Learned	2.0	2.0
E178	[86]	Experience Paper	Lessons Learned	2.0	2.0
E179	[87]	Evaluation	Framework/Method/Technique	1.0	1.0
E183	[88]	Solution Proposal	Framework/Method/Technique	1.0	3.0
E187	[89]	Evaluation	Lessons Learned	1.0	2.0
E224	[90]	Solution Proposal	Framework/Method/Technique	2.0	1.0
E230	[91]	Experience Paper	Lessons Learned	2.5	2.0

[a]Research type facet, according to [78]
[b]Contribution type facet, according to [16,17]

Finally, we discuss our findings in Section 4.4 and discuss the threats to validity in Section 4.5.

4.1. Result Overview

After the study selection, 51 papers remained in the result set. Figure 2 illustrates the publication frequency including the 3-year trend (red line) and the 5-year trend (black line). The general publication frequency shows a growing interest in the research on automotive software product lines starting in 2011.

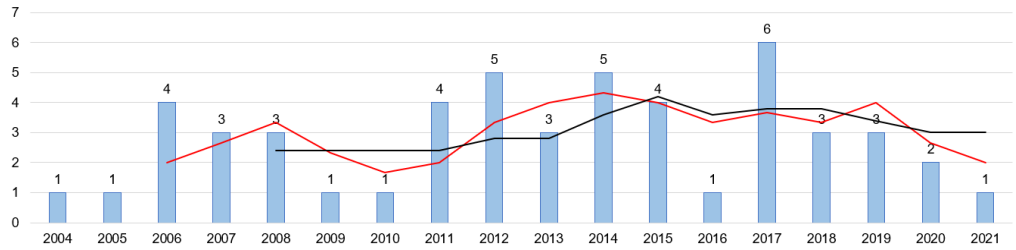

Figure 2. Overview of the publication frequency including publication trend.

Based on the initial classification from the scoping study [13], we evaluated the remaining 51 papers for their research- and contribution type classification, which is illustrated in Figure 3. This classification shows the majority of the papers contributing frameworks, methods and techniques, and that most studies are classified as *evaluation research*. It has to be noted that the papers from our result set have been filtered using the rigor-relevance model (see also Table 4). Hence, due to the selection procedure as described in Section 3.3.4, we expected mostly papers of type evaluation research, and result set contains 32 papers of type evaluation research, which we consider an indication towards practically relevant contributions. Furthermore, 12 more papers from the result set have been classified as *experience paper*, i.e., practical experience regarding automotive software product lines have been reported.

In a nutshell, we argue that our paper selection strategy resulted in a dataset that comprises those studies reporting practical relevant knowledge about automotive software product lines, which we investigate in more detail in the following sections.

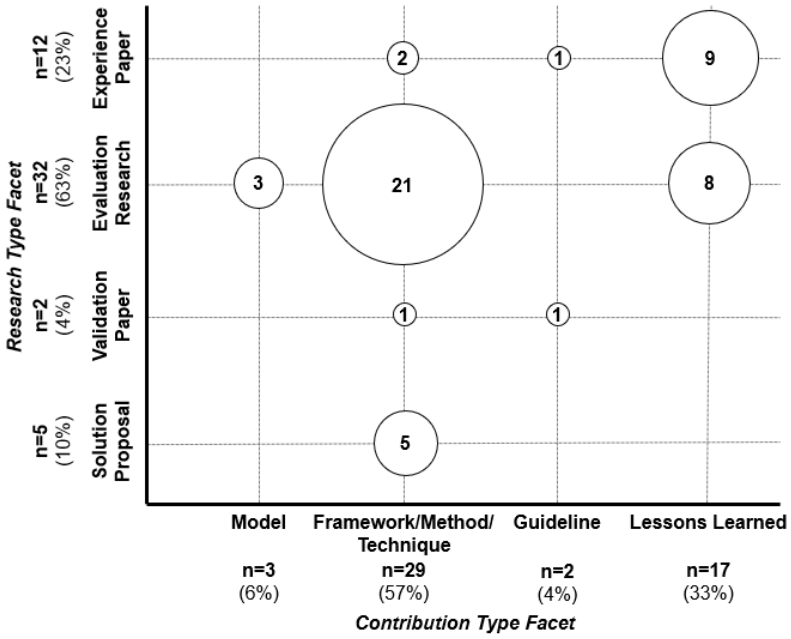

Figure 3. Classification of the result set according to research type facets and contribution type facets.

4.2. State of the Art in Holistic Approaches for Managed Evolution of Automotive Software Product Lines

In this section, we aim to answer research question RQ_1 (Table 2). As described in Section 3.4, five researchers classified the papers according to the conceptual model from Section 2.2. The classification was used to generate a mapping table for the in-depth content analysis. Based on this mapping table, Figure 4 shows the assignment of the papers to the conceptual model from Section 2.2. In Figure 4 a paper is annotated to an element if at least two researchers state that the paper makes a major contribution to that specific element. A paper can be assigned to more than one element if it contributes to several activities of the conceptual model.

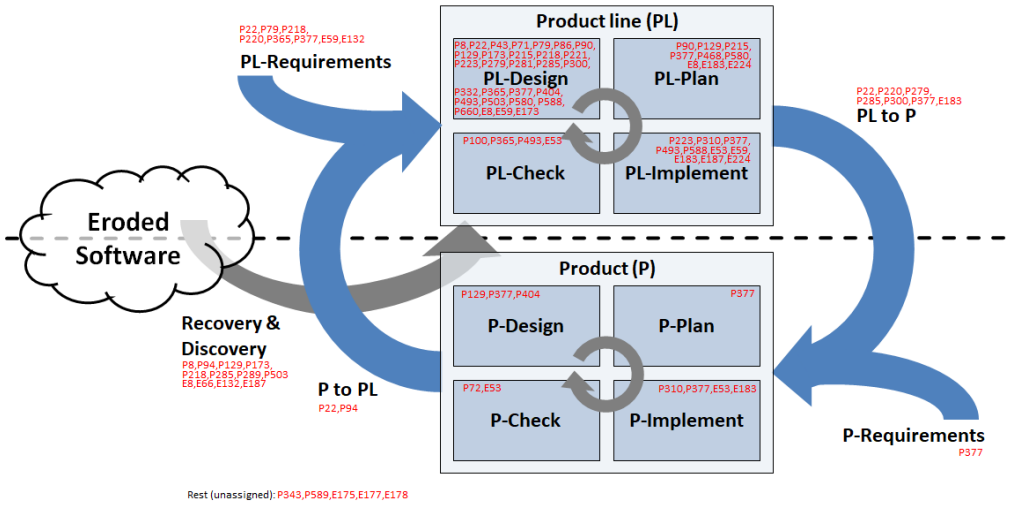

Figure 4. Assignments of the finally selected primary studies to the classification schema (Section 2.2) to prepare the in-depth content analysis.

Table 5 shows the assignments of the selected primary studies to the activities of our classification schema. According to our schema, none of the analyzed studies presents a holistic approach for managed evolution of automotive software product lines. Yet, some studies provide a certain coverage. For instance, study P377 [66] has been assigned to nine

categories thus covering three quadrants of our schema. It has also to be mentioned that this study only received a score of 1.0 in the rigor-relevance evaluation (Table 4), i.e., it has barely received the minimum score required for inclusion in the result set (cf. Section 3.3.4). The two primary studies P22 [41] and P129 [50] have been assigned four times each, and studies P218 [53], P365 [65], and P493 [69] have been assigned thrice. The remaining studies have been assigned once or twice only. In the subsequent Section 4.3 we will regard all studies proposing an "overall approach/process" in detail.

To facilitate the evaluation we count the assignments of studies to activities of the classification schema and display the percentages of counts in Table 5. In total, there are 92 assignments to the 13 activities of the classification schema, i.e., a study has been assigned on average to two activities. Two studies (P343 [64], P589 [73]) have been assigned to category "Rest (unassigned)". Here, none of the 13 activities in our classification schema has received at least two assignments for the corresponding study.

Table 5 reveals that the contribution of the studies concentrate on `PL-Design` (32%), followed by `Recovery & Discovery` (13%), `PL-Plan` (10%), `PL-Requirements` (9%), and `PL to P` (8%). Based on the accumulation of studies on the initial activities of the schema we hypothesize that the studies in question are primarily concerned with the development of new SPL rather than the long-term evolution of the product line. For example, architecture conformance checking related activities required to prevent architecture erosion in the long term is underrepresented in the studies: Only four studies (P72 [44], P100 [49], P365 [65], P493 [69]) can be found which our researchers have mapped to this appropriate fields.

Table 5: Assignments of the finally selected primary studies to the activities of the classification schema (Section 2.2) and the percentages of counts.

Activity	**Papers**	**%**
Recovery & Discovery	[40,48,50,51,53,59,60,70,75,79,82,89]	13
PL-Requirements	[41,45,53,54,65,66,77,82]	9
PL-Design	[40–43,45–47,50–53,55–59,61,63,65–67,69–72,74,75,77,83]	32
PL-Plan	[47,50,52,66,68,71,75,88,90]	10
PL-Implement	[56,62,66,69,72,76,77,88–90]	11
PL-Check	[49,65,69,76]	4
PL to P	[41,54,57,59,61,66,88]	8
P-Requirements	[66]	1
P-Design	[50,66,67]	3
P-Plan	[66]	1
P-Implement	[62,66,76,88]	4
P-Check	[44,76]	2
P to PL	[41,48]	2
Rest (unassigned)	[64,73,84–86]	

Remarkably few studies address the product specific development cycle (application engineering) and only study P72 [44] exclusively addresses the product specific development cycle. One reason for this could be that no specific methods and techniques are required for the product specific development cycle and, as in the classic SPL approach (cf. [12]), product development is carried out exclusively by derivation from the product line.

4.3. Methods and Techniques to Implement Automotive Software Product Lines

In this section, we aim to answer research question RQ_2 (Table 2). Specifically, we identify the methods and techniques used to implement automotive software product lines in general, and their application scope in particular. Table 6 provides a categorization, which is built based on the in-depth content analysis (see Table 8). Furthermore, the table

provides an overview of the different studies and how their major contributions are applied, i.e., in the scope of the product line as such and in the context of particular products derived from that product line. In the following, we first provide an overview before we provide the details structured following the categories listed in Table 6.

4.3.1. Overview of Methods and Techniques used for Automotive Software Product Lines

From the content analysis of the 51 selected primary studies (see Appendix B, Table 8), we derived 13 categories—each representing a cluster of methods and/or techniques. As already discussed in Section 4.2, on the one hand, most of the papers propose one method or technique and provide—if at all—a scoped evaluation only. On the other hand, most of the proposed methods and techniques are subject of only one paper. That is, Table 6 lists the topics of interest and, at the same time, provides some information about topics of interest in automotive software product line research, however, based on the result set, we can only derive a "fragmented" picture as several aspects are studies, yet in a fairly isolated manner. Nonetheless, the result set shows that some topics are studied from different perspectives, which indicates the variety of the subject field.

In the following sections, we discuss the different method/technique clusters individually and in more detail.

Table 6: Methods and or techniques of the papers included in this study.

No.	Methods and/or techniques	Papers	PL[a]	P[b]
1	Architecture evolution process	[46,50,52]	x	x
2	Cost/effort estimation	[43,48,51,74]	x	
3	Safety analysis	[44,71–73,90]	x	
4	Description languages	[67,76]	x	x
5	Architecture reengineering	[60]	x	
6	Model transformation	[49,56,58,59,61,79,87]	x	
7	Model-based requirements engineering	[41,45]	x	x
8	Overall approach/process	[50,53,59,66,70,75,77]	x	x
9	Reference architectures	[42,47,50,53,83]	x	x
10	SPL merging	[48,51,55,82,87]	x	
11	Testing / Verification	[62,64,88,89]	x	x
12	Variability management	[40,54,57,63,65,68,69,71–73,80,91]	x	x
13	Agile software development	[80,81,84–86,88]	x	

[a] Activities related to product line development (domain engineering) including Recovery & Discovery
[b] Activities related to product development (application engineering)

4.3.2. Architecture Evolution Process

This section discusses the studies P86 [46], P129 [50] and P215 [52], which were found addressing the architecture evolution process category—notably the maintenance and the long-term evolution of software product line architectures.

Gustavsson and Eklund [46] discuss the architects' work approach concerning the maintenance in the context of an evolving/changing *product line architecture* (PLA). Authors conducted a series of expert interviews with architects from Scania and Volvo Car. They found that the process for managing changes of the PLA is very similar and that the found processes from both companies can be mapped to a generic process, which consists of the five steps *need*, *impact analysis*, *solution*, *decision*, and *validation*. All of these findings are based on conclusions concerning the interviews and are not verified by a further case study.

The work of Lind and Heldal [50] is primarily focused on *reference architectures*, but also addresses the evolution of architectures. They distinguish between revolutionary and evolutionary architecture processes, which are both discussed in [50]. In particular,

authors argue that reference architectures have to be continuously maintained, notably, it is necessary to evaluate reference architectures continuously to identify bottlenecks, which is key to initiate refactoring and to support the architecture's evolution. Hence, Lind and Heldal present an empirically-grounded *reference architecture development process*, which contains activities that are primarily performed for evolutionary architecting, e.g., activities "Synthesize, Evaluate, and Verify & Validate Architecture". Such activities describe the architecture design, the measurement of architecture quality, and the validation against the requirements in the context of an evolving architecture.

Axelsson [52] discusses *revolutionary architecture process* (RAP) and *evolutionary architecture processes* (EAP). The specifically author focuses on the interplay of both processes and how the EAP is performed in practice. In an empirical study, reasons for changes in an architecture are discussed, complemented by a discussion of affected attributes, technical aspects involved, and decisions made. The study shows that RAP and EAP differ significantly. Also, Axelsson states that most literature mainly describes RAP, whereas EAP lack studies and evidence.

In summary, the primary studies assigned to this category use similar activities for analyzing/describing the evolution of (automotive) software product line architectures. Key is the process-support of the architecture processes, notably for architecture development and maintenance/evolution. A cross-company analysis conducted by Gustavsson and Eklund [46] did reveal similarities, which underpin the usefulness of a generalized/holistic concept.

4.3.3. Cost-/Effort Estimation

This section discusses the studies P71 [43], P94 [48], P173 [51], and P660 [74], which were found addressing the cost-/effort estimation category. A number of studies consider the cost-/effort estimation a key activity, especially in the course of planning a product line development endeavor.

Kiebusch et al. [43] propose metrics to measure the size of an automotive software product line. They argue that neither of the existing methods adequately measures the (unadjusted) size nor estimates the cost of process-oriented automotive software product lines. For this, authors propose the *Process-Family-Points* (PFP) analysis method to allow for size measurement and effort estimation.

Yoshimura et al. [48,51] address effort estimation methods with two studies: The approach proposed in [48] describes a software clone analysis approach, and the approach discussed in [51] proposes an estimation process based on the return on investment (ROI). Both approaches are integrated into a process for merge potential assessment of existing variants, i.e., how to (economically) reintegrate variants back into the product line. In [51], authors discuss an application of this approach and present lessons learned and open issues. As lessons learned they discuss, e.g., that software cloning may not be a good way to realize product line engineering, and that ROI predictions can strongly motivate the management to invest in product line engineering, and that architecture-centric clone analysis is a useful and practical approach to assess the merge potential of the existing systems. Open issues are, e.g., clone visualization, clone refactoring, and clone error reduction.

Gustavsson and Axelsson [74] provide a method of evaluating system designs with the purpose of enabling practitioners to systematically think about the future development of a system. For this, they use the *Real Options Theory* [93,94] that adds the possibility to put an economic value on the system adaptability attribute and, thus, motivates architects to also anticipate the actual value of future developments of an architecture.

In summary, the studies [43,51,74] use different methods to perform cost- and effort estimation actives in the course of planning software product line development. Furthermore, the studies provide process models for practically applying these methods, and the studies present evaluations of the respective approaches.

4.3.4. Safety Analysis

This section discusses the studies P72 [44], P580 [71], P588 [72], P589 [73], and [90] which were found addressing the safety analysis category.

Three studies address safety analysis for the development of safety-critical automotive software product lines. Rana et al. [44] address the problem of selecting the appropriate *Software Reliability Growth Model* (SRGM) of more than 100 currently existing SRGMs. Growth models are used for evaluating the maturity or release readiness of a software before its release and, respectively, for an optimal allocation of the test resources required. Rana et al. use a statistical model to identify the distribution of defects, which helps selecting an appropriate growth model. A case study conducted at Volvo Car Group is utilized to evaluate the proposed approach. They evaluate six standard distributions on defect inflow data from four large software projects and show that beta distribution provides the best fit to the defect inflow data. Second, Käßmeyer et al. [71] present an improved safety engineering approach for software product line development. Their approach provides an integrated change impact analysis by combining their approach with variability management. They apply their approach to an industrial example, a small part of an Advanced Driver Assistant System (ADAS), to illustrate the benefits. As demonstrated by the example, changes are propagated in one model for both variant management and safety engineering. Pett et al. [90] apply a risk-based change-impact analysis on an automotive architecture, combining risk-based testing, product sampling, and configuration prioritizing. They use information of the changes applied in a software update to prioritize relevant system variants for checking compatibility of an update to existing variants in the field. The approach is evaluated on five versions of a Body Comfort System.

De Oliveira et al. [72] propose an approach to support the generation of fault trees and FMEA analyses for products derived from a software product line. Their approach aims at reducing the effort required for performing safety analyses for the products. The study proposes a process model for model-based safety analysis, which starts with a product line hazard analysis, followed by a process step "Augmentation of PLA with failure logic" that describes how product line architecture design elements, i.e., product line components, can fail and how they contribute to the occurrence of hazards. Further steps include the definition of software product line configuration knowledge, product derivation, and safety assessment. In [73], de Oliveira et al. propose an approach to support the automated construction of modular product line safety cases. The approach uses different techniques like architecture failure modeling, functional failure modeling, and component failure modeling. Note, that the three studies [71–73] are also assigned to the category variability management (Section 4.3.13) in which we provide a further discussion.

4.3.5. Description Languages

This section discusses the studies P404 [67] and E53 [76], which were categorized in the description languages category. For the development of a software product line, a suitable description technique is key for supporting the specification of variability. Kim et al. [67] propose a concept to support the functional view (software behavior) in the component-based software development (CBSD). The approach introduces "signal flows" and "mode-dependent signal flows" for the specification of a component. In this regard, important information is provided at the component level to support the understanding of the software's behavior and to also understand dependencies among software components when reusing and adapting software components. The approach is proposed in the context of product line development in the automotive sector. However, the study does not provide extensive details concerning the degree of appropriateness of the approach for product line-driven development. The approach rather addresses a more general problem of software development in the automotive sector. The study [76] introduces the so called EMAB architecture description language. The EMAB meta model consists of two views on the architecture: the logical architecture (design layer) and the technical software architecture

(implementation layer). For both layers, mappings between the elements can be defined. The description language serves as a basis for an architecture conformance checking approach, also described in the study. The goal is to minimize software architecture erosion in the long-term. The benefits of the approach are demonstrated on a real world case study, brake servo unit (BSU) software system from automotive software engineering.

4.3.6. Architecture Reengineering

This section discusses the study P289 [60]), which addresses the architecture reengineering. Architecture erosion has become a major challenge in automotive software engineering, which results in a considerable effort for maintenance and software system evolution. Furthermore, architecture erosion is a major problem affecting software reuse [35]. In this context, Strasser et al. [60] propose an approach for reengineering an eroded software product line architecture. First, all relevant variation points and the associated functional requirements of a component are identified. A variability analysis is performed by the *Product Line UML-based Software Engineering* (PLUS; [95]) approach, which is also used to describe appropriate variability models. Based on the analysis results of the product line extraction process, architecture components are identified and designed in the next step. This approach can also be assigned to the recovery & discovery activity.

4.3.7. Model Transformation

This section discusses the studies P100 [49], P223 [56], P281 [58], P285 [59], P300 [61], E66 [79] and E179 [87], which were assigned to the model transformation category. Several studies utilize model transformation techniques to transform artifacts of software product line development into different models, such that techniques, e.g., for model analysis can be applied.

For instance, White et al. [49] propose an approach to debugging feature model configurations and automating configuration evolution, called CURE (Configuration Understanding and REmedy). Configurations and feature models can be transformed into constraint satisfaction problems (CSP) to automatically diagnose errors and repair invalid feature selections.

Merschen et al. [56] present a prototypical framework for the analysis of embedded software product lines. They analyze artifacts by transforming them into models, which are used in an analysis process based on model transformation languages. The automated preprocessing is implemented as model transformations in ATLAS Transformation Language (ATL) and Epsilon Transformation Language (ETL).

Leitner et al. [58] introduce EAST-ADL2 in an automotive software product line including a transformation from AUTOSAR to EAST-ADL2. Basic variability information can be automatically extracted during the transformation step. Different mapping strategies are analyzed to generate a correct model and to reduce losses in the transformation process. Furthermore, they describe the implementation of the transformation process.

Polzer et al. [59] present a framework for model-based product lines of embedded systems. The framework supports the (semi-)automated extraction of models from existing requirement-, test-, and implementation artifacts.

Wille et al. [79] propose a variability mining procedure that semi-automatically identifies variability information in a set of related models that were realized by clone-and-own approaches. They generate a delta language automatically based on the results of the variability mining. The procedure is evaluated using IBM Rational Rhapsody state charts from an SPL and MATLAB/Simulink models from an industrial case study.

Kehrbusch at al. [87] propose an automated syntactical similarity analysis for software component interfaces to support the software product line extraction and maintenance. They want to identify identical or similar components under development and to be able to extract a generic interface for the establishment of a generic component. This generic component can then be reused in the different analyzed project contexts.

Finally, Wang [61] presents a study on the application of model transformation techniques on the development of automotive software product lines. The paper aims at understanding the state-of-the-art techniques and to identify model transformation challenges in product-line-based automotive software—notably to help developers choosing the model transformation technique appropriate for the respective situation. Wang distinguishes between model transformation at the same abstraction level and model transformation across different abstraction levels. The transformations are implemented by using the tool GReAT (Graph Rewrite And Transformation). Wang presents the results of a case study with a simplified enhanced cruise control system (eCCS). As lessons learned, he states that model transformations with well-design transformation rules yield consistent implementations across vehicle variations and can thus reduce the efforts to create and maintain the design variations for different vehicles. In addition, Wang reveals that current transformation features partially meet the needs of derivative design, and still require improvement.

In summary, the studies [49,56,58,59,61,79,87] show that model transformation techniques play a key role in the development of automotive software product lines. By means of model transformations, tools and techniques can be used in the different process steps. The effort to create and maintain design variations, e.g., for different vehicles, can be reduced. Well-design transformation rules are crucial for the effectiveness of a model transformation.

4.3.8. Model-based Requirements Engineering

This section discusses the studies P22 [41] and P79 [45], which were assigned to the model-based requirements engineering category. Requirements engineering requires appropriate support by methods and techniques to capture and manage the requirements for a software product line to maintain the high degree of variability.

For this, Gleirscher et al. [41] present an approach for integrating innovation management with requirements and technology management. Requirements-based innovations are usually motivated by newly elicited requirements or needs originating from market research, whereas technology-based innovations are motivated by new or emerging technologies (e.g., specific platform components and platform services). To identify innovations, they use models of feature hierarchies, platform service hierarchies, and a platform component models. Furthermore, to accomplish innovations they define activities for requirements-based innovations and technology-based innovations.

Aoyama and Yoshino [45] present an aspect-oriented approach for the requirements engineering in software product line development. Automotive systems often deal with a wide spectrum of interwoven functional and nonfunctional requirements. They apply a multi-dimensional aspect-oriented modeling and analysis to generate multiple software product lines for automotive systems. The approach separates intersecting non-functional requirements (NFR) into primitive concerns (aspects) and introduces quantitative metrics of each NFR/aspect.

4.3.9. Overall Approach/Process

This section discusses the studies P285 [59], P377 [66], P503 [70], P129 [50], P218 [53], E8 [75], E53 [76], and E59 [77], which were assigned to the overall approaches/processes category. Several studies propose an overall approach and/or outline a process for automotive software product line development. In some cases, specific methods and techniques are proposed, such that the respective studies are also assigned to other categories in Table 6.

Polzer et al. [59] present a framework for model-based automotive software product lines in which model transformations play an important role. The framework supports the creation of context-specific views, which provide a detailed description of the domain engineering and application engineering process tasks and artifacts. The approach also

addresses the recovery and discovery of an evolved product line using a model-extraction process. Finally, authors define a product-derivation process for application engineering.

Hardung et al. [66] propose a framework to improve the reuse of automotive software. Their framework is based on the *Product Line Practice* (PLP; [11]) process model. Authors explain how to perform the modularization of the product line core assets which are contained in a function repository. The process also defines how to develop products from that function repository using a standard software core. Finally, authors list tools to support the processes.

Tischer et al. [70] present experiences regarding the introduction of product lines at Bosch Gasoline Systems. Their study outlines the different relevant areas for SPL-based development, e.g., architecture development, product-line scoping and core asset development, market-oriented development, measurement of product line success, and product quality management. Furthermore, they discuss why the product line's deployment at Bosch has been delayed.

Lind and Heldal [50] and Eklund et al. [53] both propose an architecture-centered development approach and present detailed process descriptions. One focus of both studies is on reference architectures so that the studies are discussed in detail in category *reference architectures* (Section 4.3.10).

Grewe et al. [75] propose an approach for extracting, designing, and managing architecture concepts. First, they propose methods to extract initial architectures by recovery/discovery techniques. Second, they show architecture design principles / pattern that they worked out in their automotive domain projects. In addition, they suggest techniques for measuring of architecture quality. The approach is evaluated on a real world example, a longitudinal dynamics torque coordination.

Bilic et al. [77] identify and define a Product Line Engineering (PLE) process at the engines control department of Volvo CE. For this purpose, they analyze the existing Model-based Systems Engineering activities and discuss the implications of the migration from the current development process to a Model-based PLE-oriented process. They also identify hindering factors of the current development process, e.g., ambiguous and incomplete information.

In summary, the studies of this category describe an overall development approach for automotive software product line development, but at different levels of detail and with different focal points: the studies [50] and [53] are architecture-centered and contain more detailed process descriptions. In contrast, [70] remains fairly vague and does not contain a process description; instead, it provides an experience report. [77] introduces a model-driven process using languages like OVM and SysML. The other studies [59,66] propose frameworks for software product line development: [59] focuses on model-based development using techniques like model transformation. The framework proposed in [66] allows for classifying software according to possible ways of reusing it, and it links the development process to the environment, i.e., repositories and tools. However, specific guidelines for developing automotive software product lines are not provided.

Compared to our classification schema (Section 2.2), neither approach [50,53,59,66,70, 77] covers all activities. For instance, recovery & discovery is only supported by [59]. Yet, the framework in [59] does not address a product-specific development of implementation artifacts. Instead, implementation is only performed during domain engineering and serves as the basis for product derivation.

4.3.10. Reference Architectures

This section discusses the studies P43 [42], P90 [47], P129 [50], P218 [53], and E173 [83], which were assigned to the reference architecture category. Several studies propose reference architectures as a key element for the development of software product lines.

Martínez-Fernández et al. [42] present a survey on the benefits and drawbacks of using the software reference architecture AUTOSAR, an open industry standard for the automotive software architecture between suppliers and manufacturers. They conducted

an online survey that addressed experienced AUTOSAR practitioners. They found standardization and reuse as most popular benefits, and complexity and initial investment as most remarked drawbacks and risks of using AUTOSAR.

Eklund and Bosch [47] proposed a reference architecture for embedded open software ecosystems consisting of 17 key decisions resulting in four architectural patterns. The architectural patterns are: *device abstraction*, *data and service provision*, *device and information composition*, and *safety-critical, certified and open application access*. These patterns have to be instantiated as a product line architecture for platform design. Furthermore, they define quality attributes for the reference architecture: *composability*, *deployability of new functions*, *stability over time*, *configurability*, *consistent user interface*, and *dependability*. Their approach is demonstrated with a prototypically implemented architecture that satisfies selected key decisions and quality attributes.

Lind and Heldal [50] study the research question *"How can a reference architecture for automotive Systems be developed in a component-based setting, and how is it utilized in product development to increase reuse?"*. They focus on the definition of a development process with a reference architecture as a core artifact. The process is divided into E/E architecture development and E/E systems development. Their reference architecture is developed and maintained during E/E architecture development and serves as the basis for developing a product-specific architecture for E/E systems development. The proposed development process has been validated in several steps by projects at Saab Automobile AB. The result of the validation shows that the process works well at Saab.

Eklund et al. [53] propose an architecture-centered development process and describe how reference architectures are used to improve the development process. Authors describe the design, verification, dissemination, and maintenance of the reference architecture. Based on their experiences, they conclude that dissemination and maintenance require more resources than the development of the reference architecture. The proposed process results on the experiences with architecture-centered development at Volvo Cars.

Oliinyk et al. [83] give guidance to practitioners how to structure automotive product lines and their feature models. They analyse the strengths and weaknesses of alternatives in structuring automotive product lines and their features and investigate to what degree the selected structures are realizable with existing tool support.

Summarized, the studies [50] and [53] focus on the development process and how reference architectures are embedded into the development process. In contrast, the work of Eklund and Bosch [47] focus on the particular design of a reference architecture grounded in design decisions and patterns. The study [83] give guidance how to structure automotive product lines and their feature models. The benefits of drawbacks of the AUTOSAR reference architecture are presented in [42]. As mentioned in [47], AUTOSAR assumes an integration-centric approach in which all integration work concerning software components is done by an OEM. Yet, authors of [47] argue that it is desirable to move away from integration-centric development of software towards a development in a open software ecosystem.

4.3.11. Software Product Line Merging

This section discusses the studies P94 [48], P173 [51], P221 [55], and E132 [82], which were assigned to the SPL merging category. If independently developed systems, e.g., Diesel and Gasoline software systems, share a high degree of common functionalities, merging those products into one product line is reasonable.

The two studies by Yoshimura et al. [48,51] primarily focus on effort estimation techniques for software product line merging (as already discussed in the category *cost/effort estimation*, Section 4.3.3). Both studies also outline merging strategies. However, details about the actual process of performing the merge are missing in both studies.

Tischer et al. [55] present their experiences from merging two large-scale software product line development projects (a Diesel and Gasoline software system), which was motivated by the high synergy potential. The merge was performed at the different levels:

organization, *software architecture*, *development environment*, and *processes and methods*. The study provides a comprehensive overview of the challenges and solutions chosen to merge these large-scale systems.

Ignaim and Fernandes [82] show a practical evolution-based approach to migrate and evolve a set of variants of a given product into an SPL. The approach starts with a reverse engineering phase to synthesize the feature model (FM) The FM is then upgraded and refined in the forward engineering phase. Here, the SPL is evolved by new customer requirements in an incremental way.

Kehrbusch et al. [87] extract SPL by automated syntactical similarity analysis (see category *model transformation*). The analysis supports the identification of similarities between two interfaces and can also be applied to monitor the evolution of software component interfaces.

4.3.12. Testing/Verification

This section discusses the studies P310 [62], P343 [64], E183 [88], and E187 [89], which were assigned to the testing and verification category. Due to the high degree of variability that results in a potentially huge number of products, testing and verification of a software product line are challenging tasks.

Lochau et al. [62] present an approach of pairwise testing in the SPL context by providing a mapping between feature models and a reusable test model in the form of statecharts. Therefore, they investigate the relationship between feature-based coverage criteria and model-based coverage criteria. A further contribution is a reasoning about the applicability of this approach. The approach is validated by a case study from automotive software engineering.

Scheidemann [64] proposes an incremental process using approximation algorithms for minimizing the number of configurations needed to verify the completeness of the configuration space. For this purpose, the author introduces the concept of "locality sets" containing architectural elements that realize a functionality, which is concerned with a particular requirement. The proposed algorithms either choose the minimum set of configurations necessary to verify all requirements for all configurations, or to maximize the verification coverage of the software product line as a whole by choosing the "best" configurations. A further contribution is a technique for automatically determining commonalities in architecture and requirements.

Ebert et al. [88] present methodology called TIGRE to reduce redundant testing effort by intelligent planning and reporting of the test execution. The approach consists of two phases: the initial identification and documentation of relevant data for product line testing and a second phase focusing on the usage of this data for testing activities during each agile sprint. The study also discusses the lessons learned from applying the methodology in an industrial environment.

Shahin et al. [89] propose an approach to lifting behavior alteration analysis, including configurable visualization of analysis results. They apply an existing declarative analysis to a set of automotive software product lines from General Motors. Finally, they discuss the lessons learned throughout the project.

4.3.13. Variability Management

This section discusses the studies P8 [40], P220 [54], P279 [57], P332 [63], P365 [65], P468 [68], P493 [69], P580 [71], P588 [72], P589 [73], E127 [80], and E230 [91], which were assigned to the (general) variability management category. Managing the variability within a software product line constitutes a challenging task and is addressed by nearly one third of the studies from the result set.

Jin Sun et al. [40] propose a component-based development process based on variability types. They identify several variability types that may occur in ECU-related development: variability of software components, variability of logic components, variability of sensor components, variability of actuator components, variability of setpoint genera-

tor components, variability of output device components, and variability of component interfaces. Furthermore, they propose a process for developing ECUs that focusses on managing the variability by using the introduced variability types.

Boutkova [54] present their experiences with variability management in the requirements specification process at Daimler passenger car development. Based on these experiences, Boutkova proposes a new feature-based variability management (FBVM) approach. This approach is extended by a decentralized variability modeling approach that supports variability modeling of individual components and systems as well as modeling a product as a whole.

Graf et al. [57] propose a graph-based approach for variant modeling and management. The approach focuses on managing, modeling, and combining the different kinds of knowledge in software product line development, i.e., combining local expert knowledge with domain knowledge provided by application groups, and the integration of overall management knowledge.

Brink et al. [63] propose an approach to link hardware and software variability models. Their approach distinguishes between software and hardware variants using separate variability models. They further distinguish two different kinds of dependencies between hardware- and software product lines.

Millo and Ramesh [65] propose an approach to link the design-level variability with the requirements-level variability. The requirements and designs are expressed as extended finite state machines, so called *Finite State Machines with Variability* (FSMv). The variability between designs and requirements is based on a conformance relation between design and requirements models of the software product line features. An algorithm is proposed for checking conformance between the models and is implemented on the verification tool SPIN.

Manz et at. [68] propose an approach for enabling traceability, integrated configuration of software variants, and links using integrated feature modeling. An integrated feature model considers the different development phases, abstraction layers, and development artifacts.

Kato and Yamaguchi [69] present an approach for variability management focused on pair-wise feature interactions. They accumulate the occurrence of the feature interactions for all past products using a feature interaction matrix to visualize feature interactions.

Käßmeyer et al. [71] propose a process for a model-based change impact analysis. They use methods and techniques for safety analysis (see category *safety analysis*, Section 4.3.4) and configuration/variant management, and focus on the combination of both.

De Oliveira et al. [72] propose an approach to support safety analysis for software product lines. Their approach provides guidelines regarding the use of model-based development, safety analysis, and variability management tools. In [73], de Oliveira et al. also address safety engineering in software product lines. They propose a method to support automated construction of modular product line safety cases. The method uses outputs provided by model-based development, safety analysis, and variability management tools.

Hayashi and Aoyama [80] propose a structural analysis method of variability for multiple product lines using an extended model of OVM (Orthogonal Variability Model). The approach is applied to Agile Product Line Engineering (APLE) of their automotive software systems (see category *agile software development*).

Wägemann et al. [91] conduct interviews with the automotive industry experts. They want to investigate the usefulness (or necessity) of automated PLA design space exploration for current industrial practice. The study derives a set of challenges for automated design space exploration, and distinguishes between technical issues, domain issues, and end-user issues.

Summarized, variability management is addressed by the selected studies in different ways using different methods and techniques. Several studies deal with the modeling of variability (e.g., [40,54,57,63,65,68,69,80] and with the tracing of variability information

across different levels of abstraction, like requirements and design [65], hardware and software [63], or safety analysis assets and design assets [71–73]. Three studies focus on the combination of variability management and safety analysis and are thus also classified in category *safety analysis* [71–73] (Section 4.3.4).

4.3.14. Agile Software Development

This section discusses the studies E127 [80], E130 [81], E175 [84], E177 [85], E178 [86], and E183 [88], which were assigned to the agile software development category. Agile development is also gaining importance in the area of product lines and promises many benefits.

Hayashi and Aoyama [80] (see category *variability management*) show how Software Product Line Engineering and Agile Software Development can be integrated towards Agile Product Line Engineering (APLE). After studying existing work, they see a lack of variability management in the APLE and thus address this in their study. To demonstrate their approach APLE is applied to the development of automotive system using ultrasonic sensors at DENSO. In a further study [81], Hayashi et al. propose an agile development method for multiple product lines by iteratively reusing process assets in application engineering.

The three studies by Hohl et al. [84–86] investigate the mapping of agile development to automotive software product lines: The challenges for the adoption of agile development are mainly of organizational, technical, and social nature [84]. For example, dependencies between departments and suppliers are challenging and must be considered [85]. However, agile software product line engineering is promising and can add value to existing development approaches [86].

Ebert et al. [88] apply their TIGRE method (see category *testing/verification*) in an agile development process. Here, they test the product line in agile sprints. As lessons learned, the study shows that agile software development required agile tools for data management.

Summarized, all studies show attempts how to integrate agile development into the field of automotive software product line engineering. It is also noticeable here that all studies are from the last five years.

4.4. Discussion

We conducted this review with the objective to understand the current state-of-the-art in holistic approaches for a managed evolution of software product line architecture in automotive software engineering. Furthermore, we aimed to identify the methods and techniques used to implement automotive software product lines in general, and their usage scope in particular. To achieve these objectives, we performed an in-depth review by using a refined conceptual model for automotive specific software product line development [34] as well as the results of a scoping study [13] in the field of automotive software engineering. In the following, we summarize the most important findings of this review according to the research questions.

4.4.1. RQ1: What is the current state-of-the-art in holistic approaches for managed evolution of automotive software product lines?

The in-depth review reveals that none of the studies represents a holistic approach for managed evolution of automotive software product lines according to the classification schema (Section 2.2). On average two activities of our classification schema are addressed by the selected primary studies (see Table 5). In addition, according to the results of Section 4.2 most studies refer to activities related to product line development (domain engineering) including Recovery & Discovery (86%). The analysis of the methods and techniques in Section 4.3 produces comparable results.

4.4.2. RQ2: What particular methods and techniques are used to implement a managed evolution of automotive software product lines?

From the content analysis of the primary studies, we clustered the proposed methods and techniques into 12 categories (see Table 6). Most of the papers propose one method or technique and provide—if at all—a scoped evaluation only. Based on the result set, we can only derive a "fragmented" picture as several aspects are studies, yet in a fairly isolated manner. Nonetheless, the result set shows that some topics are studied from different perspectives, which indicates the variety of the subject field.

Another finding is the accumulation of methods and techniques addressing the first activities of the schema like the product line design (see Section 4.2). This indicates that the focus is on how to build up a product line, which is called *revolutionary architecture process* (RAP) in [52]. Axelsson [52] states that most literature mainly describes RAP, whereas *evolutionary architecture processes* (EAP) lack studies and evidence (see Section 4.3). Methods and techniques required for long term evolution (i.e., EAP according to [52]) like architecture conformance checking are only addressed by few studies.

Remarkably, there is a low number of contributions concerning the product specific development cycle, which may indicate a research gap or it may indicate, e.g., that only few specific methods and techniques are required for product development within a product line based development approach. However, further developments that are separate from the product line, have to be transferred back into product line development at a later stage. The corresponding activity `P to PL` of the conceptual model was solely subject of two studies [41,48] according to the result of the categorization depicted in Table 5. But the proposed methods and techniques of both studies [41,48] have another focus (see Section 4.3) and do not support the actual objective of activity `P to PL`. Here, we hypothesis a further research gap which has to be addressed in future research towards a holistic approach for managed evolution of automotive software product line architectures.

4.5. Threats to Validity

Since the study at hand is an in-depth study grounded in a previously conducted scoping study, the dataset we relied on suffers from potential selection bias introduced by the main study. To improve the *internal validity*, we implemented a researcher triangulation for the refinement of the dataset. For the team of researchers, we built sub-teams that carried out specific tasks and the remaining researchers not involved in that very tasks performed the quality assurance activities.

5. Conclusion

We prepared the in-depth review by using a refined conceptual model for automotive specific software product line development as well as the results of a scoping study in the field of automotive software engineering. First, we aimed at collecting information about current state-of-the-art in holistic approaches for managed evolution of automotive software product lines and used our conceptual model as a reference model to evaluate the state-of-the-art. Specifically, we were interested into the coverage of the particular aspects of the conceptual model and, thus, the fields covered in current research and research gaps, respectively. The in-depth review reveals that none of the studies represents a holistic approach for managed evolution of automotive software product lines according to the used classification schema.

Next, we aimed to identify the methods and techniques used to implement automotive software product lines in general, and their usage scope in particular. We clustered the identified methods and techniques into 13 categories. On the one hand, most papers propose one method or technique and provide limited, if any, evaluation. On the other hand, most of the proposed methods and techniques are the subject of only one paper. Based on the results, we can only draw a "fragmented" picture, since several aspects are studied, but in a rather isolated way. Nevertheless, the set of results shows that some topics are studied from different angles, indicating the diversity of the subject area. A further

finding is the accumulation of studies addressing the first activities of the schema like the product line design and thus aiming at building up new software product lines. In contrast to this, approaches for long term evolution are underrepresented and should be addressed in future research.

It is evident from the study that the topic of applying agile software development to automotive software product line development has become very important in recent years. In the papers after 2016, more than every third paper addresses this topic. The results in this field are promising and it can be assumed that the results from this field will soon find their way into practice.

The small number of contributions dealing with the product-specific development cycle is noticeable, which may indicate a research gap or, for example, that only a few specific methods and techniques are required for product development within a product line-based development approach. However, there may be further developments that do not take place in the product line but at the level of the individual products. These further developments that are separate from the product line must be transferred back into the product line development at a later time. Here, we see a research direction for future work.

Furthermore, real industry examples should be increasingly used for future research as the studies show that many approaches lack practicability and applicability. It has been shown that the individual approaches always represent only a partial aspect, and integration is usually not possible as different formalizations and no overall language or model underlies them. Therefore, we see the formalization and modeling as a future research area to achieve a seamless integration of the individual approaches.

Author Contributions: Conceptualization, C.K. and A.R.; methodology, C.K.; validation, C.K., A.R., M.S., A.S., and M.V.; investigation, C.K., A.R.; resources, C.K.; writing—original draft preparation, C.K.; writing—review and editing, C.K., A.R., M.S., A.S., and M.V.; visualization, C.K.; supervision, A.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADAS	Advanced Driver Assistant System
AGR	Agreement Processes
AORE	Aspect-Oriented Requirements Engineering
ATL	ATLAS Transformation Language
CBSD	Component-based Software Development
CSP	Constraint Satisfaction Problems
CTF	Contribution Type Facet
CURE	Configuration Understanding and REmedy
DSE	Design Space Exploration
EAP	Evolutionary Architecture Processes
eCCS	Enhanced Cruise Control System
ECU	Electronic Control Unit
ETL	Epsilon Transformation Language
FBVM	Feature-based Variability Management
FMEA	Failure Mode and Effects Analysis
FSMv	Finite State Machines with Variability
GReAT	Graph Rewrite And Transformation
MBT	Model-based Testing
NFR	Non-functional Requirements
OEM	Original Equipment Manufacturer
ORG	Organizational Project-Enabling Processes
P	Product
PFP	Process-Family-Points
PL	Product Line
PLA	Product Line Architecture
PLE	Product Line Engineering
PLP	Product Line Practice
PLUS	Product Line UML-based Software Engineering
PRO	Project Processes
RAP	Revolutionary Architecture Processes
REU	Software Reuse Processes
RG	Research Question
ROI	Return on Investment
RTF	Research Type Facet
SBSE	Search Based Software Engineering
SPL	Software Product Line
SRGM	Software Reliability Growth Model
SUP	Software Support Processes

Appendix A Primary Studies

In this appendix, we provide an overview of the finally selected primary studies that form our result set. A quick overview of the studies selected can be found in Table 7.

Table 7: Overview of the primary studies selected for in-depth data analysis.

Id	Ref	Title	Summary
P8	[40]	A component-based process for developing automotive ECU software	Definition of variability types and variation points for ECUs and a component based development process for developing ECUs.
P22	[41]	A model-based approach to innovation management of automotive control systems	Model based approach to innovation management of automotive systems focusing on requirements based innovation, and on technology based innovation.
P43	[42]	A Survey on the Benefits and Drawbacks of AUTOSAR	Survey paper, which aims to gather evidence on AUTOSAR.
P71	[43]	An unadjusted size measurement of embedded software system families and its validation	Metrics to measure the size of a system family oriented software development for estimation of development costs.
P72	[44]	Analysing defect inflow distribution of automotive software projects	Approach for selecting the appropriate SRGM (Software Reliability Growth Model) model from more than 100 possible SRGMs using statistical methods.
P79	[45]	AORE (aspect-oriented requirements engineering) methodology for automotive software product lines	AORE (Aspect-Oriented Requirements Engineering) methodology to model functional and non-functional requirements of automotive software systems of multiple product lines aligned on a safety-critical distributed real-time architecture.
P86	[46]	Architecting automotive product lines: Industrial practice	Study that shows how architects work with maintaining and changing PLA.
P90	[47]	Architecture for embedded open software ecosystems	Open software ecosystems for embedded systems based on architectural cornerstones and continuous integration; reference architecture design.
P94	[48]	Assessing merge potential of existing engine control systems into a product line	Approach to assess potential to merge existing systems into a product line. Procedure for performing the migration.
P100	[49]	Automated diagnosis of feature model configurations	Constraint-based diagnostic approach for debugging errors in feature model configurations, and which can also be applied for software configuration evolution.
P129	[50]	Automotive system development using reference architectures	Development process for E/E Architectures using reference architectures: The process describes how to create and maintain the architecture and how to refine this into a product-specific architecture in product development projects.
P173	[51]	Defining a strategy to introduce a software product line using existing embedded systems	Approach to assess potential to merge existing systems into a product line.
P215	[52]	Evolutionary architecting of embedded automotive product lines: An industrial case study	Investigation of the evolutionary and revolutionary system architecture process in practice and the interplay between both.
P218	[53]	Experience of Introducing Reference Architectures in the Development of Automotive Electronic Systems	They describe the evolution of the architecture development process at Volvo Cars. Specifically, they explain the background to why they introduced an architecture centred development process and how they apply this in practice. The result of the process is a reference architecture as a basis for the design of several projects.
P220	[54]	Experience with variability management in requirement specifications	Feature based variability management (FBVM) approach: The requirements are first mapped to features. Then, the specification author defines the product variants by selecting the relevant features. For a new specification document, the set of relevant product variants now defines all necessary requirements.
P221	[55]	Experiences from a large scale software product line merger in the automotive domain	Experience report: Introduction of SPL approach at Bosch Gasoline Systems.

Id	Ref	Title	Summary
P223	[56]	Experiences of applying model-based analysis to support the development of automotive software product lines	Prototypical framework for the analysis of embedded systems product lines, including view-supported analyses of Simulink models, evolution aspects of variability, and traceability.
P279	[57]	IVaM: Implicit variant modeling and management for automotive embedded systems	Graph-based approach for the modeling and analysis of functional variants. They apply design space exploration (DSE) techniques and use knowledge based modeling approaches.
P281	[58]	Lightweight introduction of EAST-ADL2 in an automotive software product line	Introduction of EAST-ADL2 in an automotive SPL, including mapping from AUTOSAR to EAST-ADL2, as well as implementation of a single point of control with respect to variability.
P285	[59]	Managing complexity and variability of a model-based embedded software product line	Framework for model-based product lines of embedded systems, including the following methods: Creation of a model-based PL from a historically grown product family, analyze information by context-specific views, and efficient product derivation.
P289	[60]	Mastering Erosion of Software Architecture in Automotive Software Product Lines	Approach for architecture regeneration applied to a real world example.
P300	[61]	Model transformation for high-integrity software development in derivative vehicle control system design	Study of applying model transformation to high-integrity software development for derivative vehicle systems.
P310	[62]	Model-based pairwise testing for feature interaction coverage in software product line engineering	Pairwise testing in the SPL context: Mapping between feature models and a reusable test model in the form of statecharts. Investigation of the relationship between feature-based coverage criteria and model-based coverage criteria.
P332	[63]	On hardware variability and the relation to software variability	Approach to relate a hardware variability and a software variability model using properties (hierarchically ordered name-value pairs) and a tool-based matching algorithm.
P343	[64]	Optimizing the selection of representative configurations in verification of evolving product lines of distributed embedded systems	Method for determining a minimal set of configurations such that the successful verification of this small set implies the correctness of the entire product family.
P365	[65]	Relating requirement and design variabilities	Method to relate the variability from design to requirement based on a conformance relation between design and requirement models of SPL's features.
P377	[66]	Reuse of Software in Distributed Embedded Automotive Systems	Framework for the reuse of application software components for automotive manufacturers, based on the PLP process model: Process part, development of modularized software components, and product development by using a standard software core.
P404	[67]	Software behavior description of real-time embedded systems in component based software development	Concept to support the functional view (software behavior, e.g., "signal flows" and "mode dependent signal flows") at the component level.
P468	[68]	Towards integrated variant management in global software engineering: An experience report	Integrated variant management for building a common SPL by distributed teams. Parts of the method: Common development process with integrated feature model, and common configuration and revision control.
P493	[69]	Variation management for software product lines with cumulative coverage of feature interactions	Variant management with respect to pair-wise feature interaction.
P503	[70]	Why does it take that long? Establishing Product Lines in the Automotive Domain	Experience report: Introduction of SPL approach at Bosch Gasoline Systems, consisting of initial designing of the SPL, and long-term product line design and development including quality management.

Id	Ref	Title	Summary
P580	[71]	A process to support a systematic change impact analysis of variability and safety in automotive functions	Process of an integrated, model-based change impact analysis. This process is integrating model-based development, SPL engineering, and safety engineering.
P588	[72]	A model-based approach to support the automatic safety analysis of multiple product line products	Model-based approach to support the generation of safety analysis assets (fault trees and FMEA analysis) addressing multiple SPL products.
P589	[73]	Supporting the automated generation of modular product line safety cases	Approach for supporting largely automated generation of modular and reusable PL safety case architectures from the information provided by SPL feature modeling and model-based safety analysis.
P660	[74]	Evaluating flexibility in embedded automotive product lines using real options	Evaluation process for practitioners using Real Options theory that provides a way of valuing system designs and enables to think about the future in a systematic manner.
E8	[75]	Automotive Software Product Line Architecture Evolution: Extracting, Designing and Managing Architectural Concepts	A method, that combines the following techniques: architectural concept; quality measurement technique; evolutionary incremental development process. Goal: Overall development cycle for managed evolution of automotive PLAs.
E53	[76]	Control Mechanisms for Managed Evolution of Automotive Software Product Line Architectures	(1) Description language and its meta model for the specification of the software product line architecture and the software architecture of the corresponding products. Goal: Architecture conformance checking.
E59	[77]	Towards a Model-Driven Product Line Engineering Process – An Industrial Case Study	They identify and define a Product Line Engineering process that is aligned with Model-based Systems Engineering activities at the engines control department of Volvo CE; Discuss the implications of the migration from the current development process to a Model-based Product Line Engineering-oriented process.
E66	[79]	Extractive Software Product Line Engineering Using Model-Based Delta Module Generation	Procedure that uses the extracted variability information to generate a transformational delta-oriented SPL fully automatically.
E127	[80]	A Multiple Product Line Development Method Based on Variability Structure Analysis	Structural analysis method of variability for multiple product lines using an extended model of OVM. Agile application development method to refine development items according to variability dependency based on the analysis.
E130	[81]	Agile Tames Product Line Variability: An Agile Development Method for Multiple Product Lines of Automotive Software Systems	Agile development method for multiple product lines by iteratively reusing process assets in application engineering.
E132	[82]	An Industrial Case Study for Adopting Software Product Lines in Automotive Industry	A practical evolution-based approach to migrate and evolve a set of variants of a given product into an SPL.
E173	[83]	Structuring automotive product lines and feature models: an exploratory study at Opel	Guidance to practitioners how to structure automotive product lines and their feature models
E175	[84]	Real-life Challenges on Agile Software Product Lines in Automotive	Survey based on 16 semi-structured interviews from the automotive domain, an internal workshop with 40 participants and a discussion round on ESE congress 2016.
E177	[85]	Combining Agile Development and Software Product Lines in Automotive: Challenges and Recommendations	Combines the results of two previous publications and extends them by recommendations to combine agile software development and SPLs.
E178	[86]	Mapping Agility to Automotive Software Product Line Concerns	Based on previous work and two workshops, agility is mapped to software product line concerns.
E179	[87]	Interface-Based Similarity Analysis of Software Components for the Automotive Industry	An automated syntactical similarity analysis for software component interfaces is proposed to support the software product line extraction and maintenance.

Id	Ref	Title	Summary
E183	[88]	Applying Product Line Testing for the Electric Drive System	They present the TIGRE methodology. Goal: Reduce redundant testing effort by intelligent planning and reporting of the test execution.
E187	[89]	Applying Declarative Analysis to Software Product Line Models: An Industrial Study	Approach to lifting behavior alteration analysis, including configurable visualization of analysis results.
E224	[90]	Risk-Based Compatibility Analysis in Automotive Systems Engineering	Risk-based change-impact analysis to identify system variants relevant for retesting after an update.
E230	[91]	Exploring Automotive Stakeholder Requirements for Architecture Optimization Support	Interviews with the automotive industry experts. Goal: Research the usefulness (or necessity) of automated PLA design space exploration for current industrial practice.

Appendix B Data Extraction and Dataset Quality Assessment

The data extraction and the quality assessment was performed in a combined manner. For this, we developed a data sheet that was filled paper-wise. Table 8 shows the data extraction sheet. The actual outcome of the quality assessment of the papers included in the study can be taken from Table 9, respectively.

Table 8: Data extraction and study quality assessment sheet used in this study.

Category	Questions
Data Extraction	The data extraction was performed using the following key questions: 1. Which methods and or techniques are used? • Are the same methods used for domain and application engineering? • Are the methods/techniques used evaluated (see quality assessment)? 2. Is there a holistic approach? • Are the different activities (Section 2.2) fully covered? • Are there gaps? • What are the consequences for a holistic approach?
Quality Assessment	The quality assessment was carried out following the list of questions from Kitchenham and Charters [39, p. 28]. Each question was rated on a 5-point scale with: *criterion is* 1=not fulfilled at all, 2=partially fulfilled, 3=basically fulfilled, 4=fulfilled to a large extent, and 5=completely fulfilled and very well implemented/documented. Please note that the numbers for the individual questions are taken from [39, p. 28]. In Table 9, we refer to these numbers to present the quality assessment for the particular studies.

Table 9: Quality assessment of the papers included in this study.

Paper	Ref	1	1.1	3	4	5	6	7	8	9	10	11	12	13	14	15	16	18
P8	[40]	3	3	3	3	4	3	4	4	3	3	3	2	2	3	4	3	3
P22	[41]	4	4	5	3	3	3	2	3	2	3	3	3	3	4	4	3	4
P43	[42]	5	5	5	4	5	5	4	4	5	5	5	3	4	4	4	4	5
P71	[43]	5	4	4	3	4	4	3	3	3	3	3	3	3	4	4	3	4
P72	[44]	5	4	4	4	4	4	4	4	4	4	4	5	4	4	5	4	4
P79	[45]	3	3	3	3	3	2	3	3	2	3	3	2	2	3	3	3	3
P86	[46]	4	4	4	4	4	4	3	3	4	4	3	3	3	4	4	4	4
P90	[47]	5	4	4	3	3	4	3	3	4	4	4	4	3	4	4	4	4
P94	[48]	4	3	4	3	3	3	3	3	2	2	3	2	2	3	3	3	3
P100	[49]	4	3	4	4	4	5	4	4	5	4	4	4	4	5	5	4	4
P129	[50]	4	4	5	4	4	5	4	4	4	4	4	4	3	4	4	3	4
P173	[51]	4	3	4	3	3	3	3	2	2	2	3	3	2	3	3	3	3
P215	[52]	4	4	5	4	5	4	3	3	4	4	3	4	3	4	4	4	4
P218	[53]	4	4	4	3	2	3	2	2	3	2	3	4	2	3	3	2	3
P220	[54]	4	4	4	3	3	2	2	2	3	2	3	3	2	3	3	3	2
P221	[55]	4	4	3	3	3	3	2	3	2	3	3	2	3	3	3	3	3
P223	[56]	4	4	3	2	3	3	2	2	2	2	3	3	2	3	3	3	3
P279	[57]	4	3	4	3	2	3	2	2	2	2	2	3	2	3	3	2	3
P281	[58]	3	3	3	2	2	3	2	2	2	2	3	3	2	3	3	2	3
P285	[59]	4	4	4	3	3	3	3	3	3	3	3	2	2	3	4	3	3
P289	[60]	4	4	4	3	2	3	2	3	2	2	3	3	2	3	3	3	3
P300	[61]	4	3	4	3	3	3	4	3	3	3	3	2	2	3	3	3	3
P310	[62]	4	4	4	4	3	3	3	3	3	3	2	2	2	3	4	4	3
P332	[63]	4	4	4	3	4	3	3	3	2	3	3	3	2	3	4	4	3
P343	[64]	4	4	4	3	3	3	3	3	3	3	3	3	3	3	4	3	4
P365	[65]	4	4	3	3	2	3	2	3	3	3	2	3	3	3	3	3	3
P377	[66]	4	4	3	2	2	3	3	2	2	3	3	3	2	3	4	3	3
P404	[67]	4	4	4	3	3	3	3	3	2	3	2	2	2	3	4	3	3
P468	[68]	4	4	4	4	4	4	3	3	4	3	3	3	2	3	4	3	3
P493	[69]	4	4	4	3	4	4	3	3	3	4	3	3	4	4	4	4	4
P503	[70]	3	2	3	2	2	2	1	1	2	2	2	2	2	3	3	3	2
P580	[71]	4	4	3	4	4	3	3	4	3	3	3	3	3	3	4	4	4
P588	[72]	4	4	3	3	3	3	3	3	3	3	3	2	3	3	3	3	3
P589	[73]	4	4	3	3	3	3	3	3	3	3	3	2	2	3	3	3	3
P660	[74]	3	3	3	2	3	3	3	3	2	3	2	2	2	3	3	2	3
E8	[75]	4	4	3	4	3	3	4	2	2	3	3	3	4	3	3	3	4
E53	[76]	4	4	3	4	3	3	3	4	4	4	3	3	4	4	4	4	4
E59	[77]	4	3	4	3	3	3	3	3	2	2	3	3	3	3	3	3	4
E66	[79]	4	3	4	3	2	3	3	3	3	3	3	2	3	3	3	3	4
E127	[80]	3	3	4	3	2	3	3	4	3	4	3	4	3	4	3	3	4
E130	[81]	4	4	4	3	3	3	2	2	3	4	3	4	3	3	3	3	3
E132	[82]	3	3	3	3	2	3	3	4	3	3	3	2	3	2	3	3	3
E173	[83]	3	4	4	2	3	2	3	3	3	3	3	4	3	3	4	3	3
E175	[84]	4	4	5	2	3	4	3	2	3	3	3	4	3	3	4	3	3
E177	[85]	4	4	4	3	2	3	2	2	3	3	2	2	1	3	3	2	3
E178	[86]	4	4	4	2	3	2	2	3	3	2	3	4	2	3	3	3	3
E179	[87]	4	4	4	3	2	3	2	2	3	3	2	2	3	3	4	4	3
E183	[88]	4	5	4	4	4	4	3	4	3	3	3	3	2	2	3	3	3
E187	[89]	4	4	4	3	3	3	3	4	3	4	3	3	3	4	4	4	4
E224	[90]	5	4	5	4	4	4	4	4	4	4	3	4	3	4	4	4	4
E230	[91]	4	4	4	3	3	4	3	3	4	3	3	3	3	3	4	3	4

References

1. Schäuffele, J.; Zurawka, T. *Automotive Software Engineering: Grundlagen, Prozesse, Methoden und Werkzeuge effizient einsetzen*, 4nd revised edition ed.; Vieweg+Teubner, 2010.
2. Broy, M.; Pretschner, A.; Salzmann, C.; Stauner, T. Software-Intensive Systems in the Automotive Domain: Challenges for Research and Education . Technical Report 2006-01-1458, SAE Technical Paper, 2006.
3. Bernard, M.; Buckl, C.; Döricht, V.; M., F.; Fiege, L.; von Grolman, H.; Ivandic, N.; Janello, C.; Klein, C.; Kuhn, K.J.; Patzlaff, C.; Riedl, B.C.; Schätz, B.; Stanek, C. *Mehr Software (im) Wagen: Informations- und Kommunikationstechnik (IKT) als Motor der Elektromobilität der Zukunft*; fortiss GmbH, 2011.
4. Broy, M. Challenges in Automotive Software Engineering. Proceedings of the 28th International Conference on Software Engineering; ACM: New York, NY, USA, 2006; ICSE '06, pp. 33–42.
5. Schulte-Coerne, V.; Thums, A.; Quante, J. Automotive Software: Characteristics and Reengineering Challenges. *Softwaretechnik-Trends* **2009**, *29*.
6. Pretschner, A.; Broy, M.; Krüger, I.H.; Stauner, T. Software Engineering for Automotive Systems: A Roadmap. 2007 Future of Software Engineering. IEEE Computer Society, 2007, FOSE '07, pp. 55–71.
7. Thiel, S.; Babar, M.A.; Botterweck, G.; O'Brien, L. Software Product Lines in Automotive Systems Engineering. *SAE international journal of passenger cars-electronic and electrical systems* **2008**, *1*, 531–543.
8. Macala, R.R.; Stuckey, L.D.; Gross, D.C. Managing Domain-specific, Product-line Development. *IEEE Software* **1996**, *13*, 57–67.
9. Meyer, M.H.; Lehnerd, A.P. *The Power of Product Platforms*; Free Press: New York, NY, 1997.
10. Weiss, D.M.; Lai, C.T.R. *Software Product-line Engineering: A Family-based Software Development Process*; Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 1999.
11. Clements, P.; Northrop, L. *Software Product Lines: Practices and Patterns*; Addison Wesley, 2001.
12. Pohl, K.; Böckle, G.; Linden, F.J.v.d. *Software Product Line Engineering: Foundations, Principles and Techniques*; Springer-Verlag, 2005.
13. Haghighatkhah, A.; Banijamali, A.; Pakanen, O.P.; Oivo, M.; Kuvaja, P. Automotive software engineering: A systematic mapping study. *Journal of Systems and Software* **2017**.
14. Raatikainen, M.; Tiihonen, J.; Männistö, T. Software product lines and variability modeling: A tertiary study. *Journal of Systems and Software* **2019**, *149*, 485–510. doi:10.1016/j.jss.2018.12.027.
15. Kitchenham, B.A.; Budgen, D.; Brereton, P. *Evidence-Based Software Engineering and Systematic Reviews*; CRC Press, 2015.
16. Petersen, K.; Vakkalanka, S.; Kuzniarz, L. Guidelines for Conducting Systematic Mapping Studies in Software Engineering: An Update. *Inf. Softw. Technol.* **2015**, *64*, 1–18.
17. Petersen, K.; Feldt, R.; Mujtaba, S.; Mattson, M. Systematic mapping studies in software engineering. International Conference on Evaluation and Assessment in Software Engineering. ACM, 2008, pp. 68–77.
18. Clarke, S.; Fitzgerald, B.; Nixon, P.; Pohl, K.; Ryan, K.; Sinclair, D.; Thiel, S. The Role of Software Engineering in Future Automotive Systems Development. *SAE International Journal of Passenger Cars-Electronic and Electrical Systems* **2008**, *1*, 544–552.
19. Grimm, K. Software technology in an automotive company: major challenges. Proceedings of the 25th international conference on Software Engineering. IEEE Computer Society, 2003, pp. 498–503.
20. Gruszczynski, B. An overview of the current state of software engineering in embedded automotive electronics. Electro/information Technology, 2006 IEEE International Conference on. IEEE, 2006, pp. 377–381.
21. Fabbrini, F.; Fusani, M.; Lami, G.; Sivera, E. Software engineering in the european automotive industry: Achievements and challenges. Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International. IEEE, 2008, pp. 1039–1044.
22. Antinyan, V. Revealing the complexity of automotive software. Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering; Devanbu, P., Ed.; Association for Computing Machinery: New York,NY,United States, 2020; ACM Digital Library, pp. 1525–1528. doi:10.1145/3368089.3417038.
23. Schmid, K.; Rabiser, R.; Grünbacher, P. A Comparison of Decision Modeling Approaches in Product Lines. Proceedings of the 5th International Workshop on Variability Modeling of Software-intensive Systems (VaMoS'11); Heymans, P.; Czarnecki, K.; Eisenecker, U.W., Eds. ACM, 2011, pp. 119–126.

24. Czarnecki, K.; Grünbacher, P.; Rabiser, R.; Schmid, K.; Wasowski, A. Cool Features and Tough Decisions: A Comparison of Variability Modeling Approaches. Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems; ACM: New York, NY, USA, 2012; VaMoS '12, pp. 173–182.
25. Harman, M.; Jia, Y.; Krinke, J.; Langdon, W.B.; Petke, J.; Zhang, Y. Search Based Software Engineering for Software Product Line Engineering: A Survey and Directions for Future Work. Proceedings of the 18th International Software Product Line Conference - Volume 1; ACM: New York, NY, USA, 2014; SPLC '14, pp. 5–18. doi:10.1145/2648511.2648513.
26. Engström, E.; Runeson, P. Software product line testing–A systematic mapping study. *Information and Software Technology* **2011**, *53*, 2–13.
27. Lee, J.; Kang, S.; Lee, D. A Survey on Software Product Line Testing. Proceedings of the 16th International Software Product Line Conference-Volume 1. ACM, 2012, pp. 31–40.
28. Oster, S.; Wübbeke, A.; Engels, G.; Schürr, A. A Survey of Model-Based Software Product Lines Testing. *Model-Based Testing for Embedded Systems* **2011**, pp. 338–381.
29. Thüm, T.; Apel, S.; Kästner, C.; Schaefer, I.; Saake, G. A Classification and Survey of Analysis Strategies for Software Product Lines. *ACM Computing Surveys (CSUR)* **2014**, *47*, 6.
30. Chen, L.; Ali Babar, M.; Ali, N. Variability Management in Software Product Lines: A Systematic Review. Proceedings of the 13th International Software Product Line Conference; Carnegie Mellon University: Pittsburgh, PA, USA, 2009; SPLC '09, pp. 81–90.
31. Schobbens, P.Y.; Heymans, P.; Trigaux, J.C. Feature Diagrams: A Survey and a Formal Semantics. Requirements Engineering, 14th IEEE international conference. IEEE, 2006, pp. 139–148.
32. Chacón-Luna, A.E.; Gutiérrez, A.M.; Galindo, J.A.; Benavides, D. Empirical software product line engineering: A systematic literature review. *Information and Software Technology* **2020**, *128*, 106389. doi:10.1016/j.infsof.2020.106389.
33. Marques, M.; Simmonds, J.; Rossel, P.O.; Bastarrica, M.C. Software product line evolution: A systematic literature review. *Information and Software Technology* **2019**, *105*, 190–208. doi:10.1016/j.infsof.2018.08.014.
34. Knieke, C.; Körner, M.; Rausch, A.; Schindler, M.; Strasser, A.; Vogel, M. A Holistic Approach for Managed Evolution of Automotive Software Product Line Architectures. Special Track: Managed Adaptive Automotive Product Line Development (MAAPL), along with ADAPTIVE 2017. IARIA XPS Press, 2017, pp. 43–52.
35. Cool, B.; Knieke, C.; Rausch, A.; Schindler, M.; Strasser, A.; Vogel, M.; Brox, O.; Jauns-Seyfried, S. From Product Architectures to a Managed Automotive Software Product Line Architecture. Proceedings of the 31st Annual ACM Symposium on Applied Computing; ACM: New York, NY, USA, 2016; SAC'16, pp. 1350–1353. doi:10.1145/2851613.2851964.
36. de Silva, L.; Balasubramaniam, D. Controlling Software Architecture Erosion: A Survey. *Journal of Systems and Software* **2012**, *85*, 132–151.
37. Kuhrmann, M.; Fernández, D.M.; Daneva, M. On the pragmatic design of literature studies in software engineering: an experience-based guideline. *Empirical Software Engineering* **2017**. doi:10.1007/s10664-016-9492-y.
38. Ivarsson, M.; Gorschek, T. A method for evaluating rigor and industrial relevance of technology evaluations. *Empirical Software Engineering* **2011**, *16*, 365–395. doi:10.1007/s10664-010-9146-4.
39. Kitchenham, B.; Charters, S. Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE-2007-01, Keele University, 2007.
40. Her, J.S.; Choi, S.W.; Cheun, D.W.; Bae, J.S.; Kim, S.D. A Component-Based Process for Developing Automotive ECU Software. *Lecture Notes in Computer Science* **2007**, *4589*, 358.
41. Gleirscher, M.; Vogelsang, A.; Fuhrmann, S. A Model-Based Approach to Innovation Management of Automotive Control Systems. Software Product Management (IWSPM), 2014 IEEE IWSPM 8th International Workshop on. IEEE, 2014, pp. 1–10.
42. Martínez-Fernández, S.; Ayala, C.P.; Franch, X.; Nakagawa, E.Y. A Survey on the Benefits and Drawbacks of AUTOSAR. Proceedings of the First International Workshop on Automotive Software Architecture. ACM, 2015, pp. 19–26.
43. Kiebusch, S.; Franczyk, B.; Speck, A. An Unadjusted Size Measurement of Embedded Software System Families and its Validation. *Software Process: Improvement and Practice* **2006**, *11*, 435–446.
44. Rana, R.; Staron, M.; Berger, C.; Hansson, J.; Nilsson, M. Analysing defect inflow distribution of automotive software projects. Proceedings of the 10th International Conference on Predictive Models in Software Engineering. ACM, 2014, pp. 22–31.

45. Aoyama, M.; Yoshino, A. AORE (Aspect-Oriented Requirements Engineering) Methodology for Automotive Software Product Lines. Software Engineering Conference, 2008. APSEC'08. 15th Asia-Pacific. IEEE, 2008, pp. 203–210.
46. Gustavsson, H.; Eklund, U. Architecting Automotive Product Lines: Industrial Practice. *Software Product Lines: Going Beyond* **2010**, pp. 92–105.
47. Eklund, U.; Bosch, J. Architecture for Embedded Open Software Ecosystems. *Journal of Systems and Software* **2014**, *92*, 128–142.
48. Yoshimura, K.; Ganesan, D.; Muthig, D. Assessing Merge Potential of Existing Engine Control Systems into a Product Line. Proceedings of the 2006 international workshop on Software engineering for automotive systems. ACM, 2006, pp. 61–67.
49. White, J.; Benavides, D.; Schmidt, D.; Trinidad, P.; Dougherty, B.; Ruiz-Cortés, A. Automated Diagnosis of Feature Model Configurations. *Journal of Systems and Software* **2010**, *83*, 1094–1107.
50. Lind, K.; Heldal, R. Automotive System Development using Reference Architectures. Software Engineering Workshop (SEW), 2012 35th Annual IEEE. IEEE, 2012, pp. 42–51.
51. Yoshimura, K.; Ganesan, D.; Muthig, D. Defining a Strategy to Introduce a Software Product Line Using Existing Embedded Systems. Proceedings of the 6th ACM & IEEE International conference on Embedded software. ACM, 2006, pp. 63–72.
52. Axelsson, J. Evolutionary Architecting of Embedded Automotive Product Lines: An Industrial Case Study. Software Architecture, 2009 & European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference on. IEEE, 2009, pp. 101–110.
53. Eklund, U.; Askerdal, O.; Granholm, J.; Alminger, A.; Axelsson, J. Experience of Introducing Reference Architectures in the Development of Automotive Electronic Systems. Proceedings of the Second International Workshop on Software Engineering for Automotive Systems; ACM: New York, NY, USA, 2005; SEAS '05, pp. 1–6. doi:10.1145/1082983.1083195.
54. Boutkova, E. Experience with Variability Management in Requirement Specifications. Software Product Line Conference (SPLC), 2011 15th International. IEEE, 2011, pp. 303–312.
55. Tischer, C.; Muller, A.; Mandl, T.; Krause, R. Experiences from a Large Scale Software Product Line Merger in the Automotive Domain. Software Product Line Conference (SPLC), 2011 15th International. IEEE, 2011, pp. 267–276.
56. Merschen, D.; Polzer, A.; Botterweck, G.; Kowalewski, S. Experiences of Applying Model-based Analysis to Support the Development of Automotive Software Product Lines. Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems. ACM, 2011, pp. 141–150.
57. Graf, S.; Glaß, M.; Wintermann, D.; Teich, J.; Lauer, C. IVaM: Implicit Variant Modeling and Management for Automotive Embedded Systems. Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2013 International Conference on. IEEE, 2013, pp. 1–10.
58. Leitner, A.; Kajtazovic, N.; Mader, R.; Kreiner, C.; Steger, C.; Weiß, R. Lightweight introduction of EAST-ADL2 in an automotive software product line. System Science (HICSS), 2012 45th Hawaii International Conference on. IEEE, 2012, pp. 5526–5535.
59. Polzer, A.; Merschen, D.; Botterweck, G.; Pleuss, A.; Thomas, J.; Hedenetz, B.; Kowalewski, S. Managing complexity and variability of a model-based embedded software product line. *Innovations in Systems and Software Engineering* **2012**, *8*, 35–49.
60. Strasser, A.; Cool, B.; Gernert, C.; Knieke, C.; Körner, M.; Niebuhr, D.; Peters, H.; Rausch, A.; Brox, O.; Jauns-Seyfried, S.; Jelden, H.; Klie, S.; Krämer, M. Mastering Erosion of Software Architecture in Automotive Software Product Lines. SOFSEM 2014: Theory and Practice of Computer Science; Geffert, V.; Preneel, B.; Rovan, B.; Stuller, J.; Tjoa, A.M., Eds. Springer, 2014, Vol. 8327, *LNCS*, pp. 491–502.
61. Wang, S. Model Transformation for High-Integrity Software Development in Derivative Vehicle Control System Design. High Assurance Systems Engineering Symposium, 2007. HASE'07. 10th IEEE. IEEE, 2007, pp. 227–234.
62. Lochau, M.; Oster, S.; Goltz, U.; Schürr, A. Model-based pairwise testing for feature interaction coverage in software product line engineering. *Software Quality Journal* **2012**, *20*, 567–604.
63. Brink, C.; Kamsties, E.; Peters, M.; Sachweh, S. On Hardware Variability and the Relation to Software Variability. Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on. IEEE, 2014, pp. 352–355.
64. Scheidemann, K.D. Optimizing the Selection of Representative Configurations in Verification of Evolving Product Lines of Distributed Embedded Systems. Software Product Line Conference, 2006 10th International. IEEE, 2006, pp. 75–84.

65. Millo, J.V.; Ramesh, S. Relating Requirement and Design Variabilities. Software Engineering Conference (APSEC), 2012 19th Asia-Pacific. IEEE, 2012, Vol. 2, pp. 35–42.
66. Hardung, B.; Kölzow, T.; Krüger, A. Reuse of Software in Distributed Embedded Automotive Systems. Proceedings of the 4th ACM international conference on Embedded software. ACM, 2004, pp. 203–210.
67. Kim, J.E.; Kapoor, R.; Herrmann, M.; Haerdtlein, J.; Grzeschniok, F.; Lutz, P. Software Behavior Description of Real-Time Embedded Systems in Component Based Software Development. Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on. IEEE, 2008, pp. 307–311.
68. Manz, C.; Stupperich, M.; Reichert, M. Towards Integrated Variant Management in Global Software Engineering: An Experience Report. Global Software Engineering (ICGSE), 2013 IEEE 8th International Conference on. IEEE, 2013, pp. 168–172.
69. Kato, S.; Yamaguchi, N. Variation Management for Software Product Lines with Cumulative Coverage of Feature Interactions. Software Product Line Conference (SPLC), 2011 15th International. IEEE, 2011, pp. 140–149.
70. Tischer, C.; Muller, A.; Ketterer, M.; Geyer, L. Why does it take that long? Establishing Product Lines in the Automotive Domain. Software Product Line Conference, 2007. SPLC 2007. 11th International. IEEE, 2007, pp. 269–274.
71. Käßmeyer, M.; Schulze, M.; Schurius, M. A process to support a systematic change impact analysis of variability and safety in automotive functions. Proceedings of the 19th International Conference on Software Product Line. ACM, 2015, pp. 235–244.
72. de Oliveira, A.L.; Braga, R.T.; Masiero, P.C.; Papadopoulos, Y.; Habli, I.; Kelly, T. A Model-Based Approach to Support the Automatic Safety Analysis of Multiple Product Line Products. Computing Systems Engineering (SBESC), 2014 Brazilian Symposium on. IEEE, 2014, pp. 7–12.
73. de Oliveira, A.L.; Braga, R.T.; Masiero, P.C.; Papadopoulos, Y.; Habli, I.; Kelly, T. Supporting the Automated Generation of Modular Product Line Safety Cases. In *Theory and Engineering of Complex Systems and Dependability*; Springer, 2015; pp. 319–330.
74. Gustavsson, H.; Axelsson, J. Evaluating Flexibility in Embedded Automotive Product Lines Using Real Options. Software Product Line Conference, 2008. SPLC'08. 12th International. IEEE, 2008, pp. 235–242.
75. Grewe, A.; Knieke, C.; Körner, M.; Rausch, A.; Schindler, M.; Strasser, A.; Vogel, M.; (Keine Angabe). Automotive Software Product Line Architecture Evolution: Extracting, Designing and Managing Architectural Concepts. In *International Journal on Advances in Intelligent Systems*; Hans-Werner Sehring., Ed.; IARIA, 2017; pp. 203–222.
76. Knieke, C.; Körner, M.; Rausch, A.; Schindler, M.; Strasser, A.; Vogel, M. Control Mechanisms for Managed Evolution of Automotive Software Product Line Architectures. *International Journal On Advances in Software* **2017**, pp. 191–210.
77. Bilic, D.; Sundmark, D.; Afzal, W.; Wallin, P.; Causevic, A.; Amlinger, C.; Barkah, D. Towards a Model-Driven Product Line Engineering Process. Proceedings of the 13th Innovations in Software Engineering Conference on Formerly known as India Software Engineering Conference; Jain, S., Ed.; Association for Computing Machinery: New York,NY,United States, 2020; ACM Digital Library, pp. 1–11. doi:10.1145/3385032.3385043.
78. Wieringa, R.; Maiden, N.; Mead, N.; Rolland, C. Requirements Engineering Paper Classification and Evaluation Criteria: A Proposal and a Discussion. *Requirements Engineering* **2005**, *11*, 102–107. doi:10.1007/s00766-005-0021-6.
79. Wille, D.; Runge, T.; Seidl, C.; Schulze, S. Extractive software product line engineering using model-based delta module generation. Proceedings of the Eleventh International Workshop on Variability Modelling of Software-intensive Systems; Schaefer, I., Ed.; ACM: New York, NY, 2017; ACM Digital Library, pp. 36–43. doi:10.1145/3023956.3023957.
80. Hayashi, K.; Aoyama, M. A multiple product line development method based on variability structure analysis. Proceedings of the 22nd International Systems and Software Product Line Conference - Volume 1; Berger, T., Ed.; ACM: New York, NY, 2018; ACM Other conferences, pp. 160–169. doi:10.1145/3233027.3233048.
81. Hayashi, K.; Aoyama, M.; Kobata, K. Agile Tames Product Line Variability. Proceedings of the 21st International Systems and Software Product Line Conference - Volume A; Cohen, M., Ed.; ACM: New York, NY, 2017; ACM Digital Library, pp. 180–189. doi:10.1145/3106195.3106221.
82. Ignaim, K.; Fernandes, J.M. An industrial case study for adopting software product lines in automotive industry. Proceedings of the 23rd International Systems and Software Product Line

Conference volume B - SPLC '19; Salinesi, C.; Ziadi, T., Eds.; ACM Press: New York, New York, USA, 2019; pp. 1–8. doi:10.1145/3307630.3342409.
83. Oliinyk, O.; Petersen, K.; Schoelzke, M.; Becker, M.; Schneickert, S. Structuring automotive product lines and feature models: an exploratory study at Opel. *Requirements Engineering* **2017**, *22*, 105–135.
84. Hohl, P.; Münch, J.; Schneider, K.; Stupperich, M. Real-Life Challenges on Agile Software Product Lines in Automotiv. International Conference on Product-Focused Software Process Improvement. Springer, 2017, pp. 28–36.
85. Hohl, P.; Stupperich, M.; Munch, J.; Schneider, K. Combining Agile Development and Software Product Lines in Automotive: Challenges and Recommendations. 2018 International Conference on Development and Application Systems (DAS); IEEE: Piscataway, NJ, 2018; pp. 1–10. doi:10.1109/ICE.2018.8436277.
86. Hohl, P.; Theobald, S.; Becker, M.; Stupperich, M.; Münch, J. Mapping Agility to Automotive Software Product Line Concerns. In *Product-focused software process improvement*; Kuhrmann, M., Ed.; Springer: Cham, 2018; Vol. 11271, *Lecture Notes in Computer Science*, pp. 409–421. doi: 10.1007/978-3-030-03673-732.
87. Kehrbusch, P.; Richenhagen, J.; Rumpe, B.; Schloßer, A.; Schulze, C. Interface-based similarity analysis of software components for the automotive industry. Proceedings of the 20th International Systems and Software Product Line Conference; Mei, H., Ed.; ACM: New York, NY, 2016; ACM Digital Library, pp. 99–108. doi:10.1145/2934466.2934468.
88. Ebert, R.; Jolianis, J.; Kriebel, S.; Markthaler, M.; Pruenster, B.; Rumpe, B.; Salman, K.S. Applying Product Line Testing for the Electric Drive System. Proceedings of the 23rd International Systems and Software Product Line Conference - Volume A; Berger, T., Ed.; Association for Computing Machinery: New York,NY,United States, 2019; ACM Digital Library, pp. 14–24. doi: 10.1145/3336294.3336318.
89. Shahin, R.; Hackman, R.; Toledo, R.; Ramesh, S.; Atlee, J.M.; Chechik, M. Applying Declarative Analysis to Software Product Line Models: An Industrial Study. 2021 24th International Conference on Model Driven Engineering Languages and Systems; IEEE: Piscataway, NJ, 2021; pp. 145–155. doi:10.1109/MODELS50736.2021.00023.
90. Pett, T.; Eichhorn, D.; Schaefer, I. Risk-based compatibility analysis in automotive systems engineering. Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings; Guerra, E., Ed.; Association for Computing Machinery: New York,NY,United States, 2020; ACM Digital Library, pp. 1–10. doi: 10.1145/3417990.3421263.
91. Wägemann, T.; Tavakoli Kolagari, R.; Schmid, K. Exploring Automotive Stakeholder Requirements for Architecture Optimization Support. 2019 IEEE International Conference on Software Architecture companion; IEEE: Piscataway, NJ, 2019; pp. 37–44. doi:10.1109/ICSA-C.2019.00015.
92. Fleiss, J.L. Measuring nominal scale agreement among many raters. *Psychological Bulletin* **1971**, *76*, 378–382.
93. Amram, M.; Kulatilaka, N. *Real options*; Harvard Business School Press Boston, Massachusetts, 1999.
94. Copeland, T.; Antikarov, V. *Real Options: A Practitioner's Guide*; TEXERE New York, 2001.
95. Gomaa, H. *Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures*; Addison-Wesley Professional, 2004.