applied sciences

MDPI

Article

Variability Management in Dynamic Software Product Lines for Self-Adaptive Systems—A Systematic Mapping

Oscar Aguayo and Samuel Sepúlveda *

Departamento de Ciencias de la Computación e Informática, Centro de Estudios en Ingeniería de Software, Universidad de La Frontera, Temuco 4811230, Chile
* Correspondence: samuel.sepulveda@ufrontera.cl

Abstract: Context: Dynamic software product lines (DSPLs) have considerably increased their adoption for variability management for self-adaptive systems. The most widely used models for managing the variability of DSPLs are the MAPE-K control loop and context-aware feature models (CFMs). Aim: In this paper, we review and synthesize evidence of using variability constraint approaches, methodologies, and challenges for DSPL. Method: We conducted a systematic mapping, including three research questions. This study included 84 papers published from 2010 to 2021. Results: The main results show that open-dynamic variability shows a presence in 57.1% of the selected papers, and on the other hand, closed-dynamic variability appears in 38.1%. The most commonly used methodology for managing a DSPL environment is based on proprietary architectures (60.7%), where the use of CFMs predominates. For open-dynamic variability approaches, the MAPE-K control loop is mainly used. The main challenges in DSPL management are based on techniques (28.6%) and open variation (21.4%). Conclusions: Open-dynamic variability has prevailed over the years as the primary approach to managing variability in DSPL, where its primary methodology is the MAPE-K control loop. Response RQ3 requires further review.

Keywords: dynamic software product lines; self-adaptive systems; runtime variability; reconfiguration; systematic mapping

Citation: Aguayo, O.; Sepúlveda, S. Variability Management in Dynamic Software Product Lines for Self-Adaptive Systems—A Systematic Mapping. *Appl. Sci.* **2022**, *12*, 10240. https://doi.org/10.3390/app122010240

Academic Editor: Vito Conforti

Received: 19 August 2022
Accepted: 29 September 2022
Published: 12 October 2022

1. Introduction

The software constantly evolves due to constant technological changes and diversification in their needs, either from the customer or the execution environment. This evolution leads to self-adaptive software systems, whose main feature is adapting the system to the different needs at runtime, either through customer requirements or the environment [1]. In [2], the authors state that facing the challenges associated with changes in runtime system variability has led to the development of various approaches to adapt to changing needs. These approaches include the self-adaptive, agent-based, autonomous, emergent, and bio-inspired systems. Dynamic software product lines (DSPL) are one approach to meet these needs by providing a conceptual framework for managing variability in such systems, generating runtime variability changes, and managing system change according to the needs of the problem context [3].

DSPLs are usually not concerned with pre-runtime system variability, allowing mixed approaches, incorporating points of variation related to the environment's static properties prior to runtime and others related to dynamic properties at runtime [4]. Therefore, variability management throughout the DSPL lifecycle is a central task, where the user, application, or generic middleware can perform these tasks manually or automatically [2]. Schmid and Eichelberger grouped several challenges related to DSPL variability [5]. These challenges include detection of feature removal for DSPL approaches, reconfiguration of components, state transfer derived from reconfigurations, the adaptation of interfaces, management of processing contexts, and handling references that are no longer available. Therefore, studying runtime variability through reconfigurations becomes an important issue when it is

required to correctly design, build and produce software artifacts using the DSPL approach. Quinton et al. [3] mention similar challenges associated with managing runtime variability, such as the need to support evolution regardless of the domain, implementation technique, or modeling approach in DSPL. Securing consistency of the models and the running system to manage runtime variability and support evolution activated by the running system or model, i.e., if the system or model changes its variability, it must be visualized.

Over the years, several methodologies have emerged to manage system variability using the DSPL approach, which can be grouped into three major groups [3]. The first approach is related to changes in system customization manually, e.g., updating a software code base. The second approach allows changes to be made manually, while the adaptation process is automated, e.g., through a DevOps process. The third approach presents an autonomous execution environment that automatically allows variability changes and subsequent reconfigurations, providing an optimal execution environment for self-adaptive systems.

Variability management in self-adaptive systems using DSPL typically manages adaptation using a MAPE-K approach [3]. This approach provides a framework for logical adaptation of a system based on four phases *Monitoring, Analysis, Planning, and Execution* that are executed in a particular order and access a shared *Knowledge* component [6]. Several problems with this approach are presented in [7], such as the need for significant training data and low initial performance due to the online learning of the approach. These problems are similar to those encountered in machine-learning and a massive amount of data, such as the problems reported in using machine-learning in agricultural big data [8].

This work aims to collect several proposed approaches, methodologies, or design patterns for managing runtime variability through software reconfigurations in DSPL for self-adaptive systems present in the literature. It will seek to analyze various proposals of constraints in the reconfiguration process, main errors during the system adaptation process, or also some notion on how to maintain the stability and reliability of the system during the evolution of variability, ensuring the consistency of the software concerning the models and the system in operation [3]. The research will address the challenges mentioned by [5], specifically variability management through component reconfiguration. For this purpose, a systematic mapping study (SMS) for the most relevant articles of the last 11 years will be carried out. This SMS pretends to collect as much data as possible on the proposals to analyze the methodological approach used to manage runtime variability by reconfiguring the solution domain of software in an execution environment.

We expect the results of this systematic mapping contribute to the research-oriented scientific community of SPLs and, eventually, DSPLs, synthesizing what has happened during the last 11 years in the evolution of runtime reconfiguration approaches for DSPLs in the context of self-adaptive systems, both at the theoretical-practical and bibliometric levels. The research questions (RQs) corresponding to this systematic mapping study are as follows:

RQ1: What approach was used to apply constraints during software reconfigurations in DSPL?
RQ2: What methodologies are currently used to manage DSPL variability during reconfigurations?
RQ3: What are the current challenges in the management of DSPL?

Answering RQ1 will allow us to catalog the main approaches for managing runtime variability through software reconfigurations. In this sense, we want to understand how DSPL variability is currently managed, e.g., whether constraint languages, reasoning engines, and optimization models, among others, are occupied. RQ2 will provide us with information on the execution environments of DSPLs, analyzing, for example, whether proprietary architectures, execution environments under some autonomous control loop, or third-party software are proposed. RQ3 will focus on the search and analysis of the main constraints in developing a DSPL, whose primary focus is runtime variability.

The study will be conducted between 2010 and 2021 since the systematic mapping study by Guedes et al. examines variability management for DSPL from 2006 to 2015 [9]. The authors expose that the area of runtime variability development is not explored in-depth, concluding that most articles do not address the reconfiguration process. Furthermore, in the bibliometric analysis of the articles, an increase in the number of articles is obtained from 2010 onwards.

The remainder of this paper is structured as follows. We present the background of the DSPLs and their context in Section 2. Section 3 presents some related work relative to SPLs or DSPLs. Section 4 presents the methodology for the SMS. Sections 5 and 6 present the results and discussion, respectively. Finally, Section 7 presents the conclusions and future work.

2. Background

The contents of this section report on the theoretical framework of SPLs and their variability. Also, DSPLs and their runtime variability are defined. Finally, self-adaptive systems are presented.

2.1. Software Product Lines

SPLs are defined as a software production methodology in which a set of software systems share common characteristics for a specific domain in a defined process [10]. Software development through SPLs is divided into two stages, *domain engineering* and *application engineering* [11]. The *domain engineering* stage refers to activities that concern, among others, the identification of commonalities and differences between product family members and the implementation of a set of shared software artifacts (e.g., components). The *application engineering* stage refers to activities that concern product derivation, i.e., the construction of individual products using a subset of the shared software artifacts [12]. Figure 1 shows the SPL framework, including the domain and application engineering stages and their respective interactions.

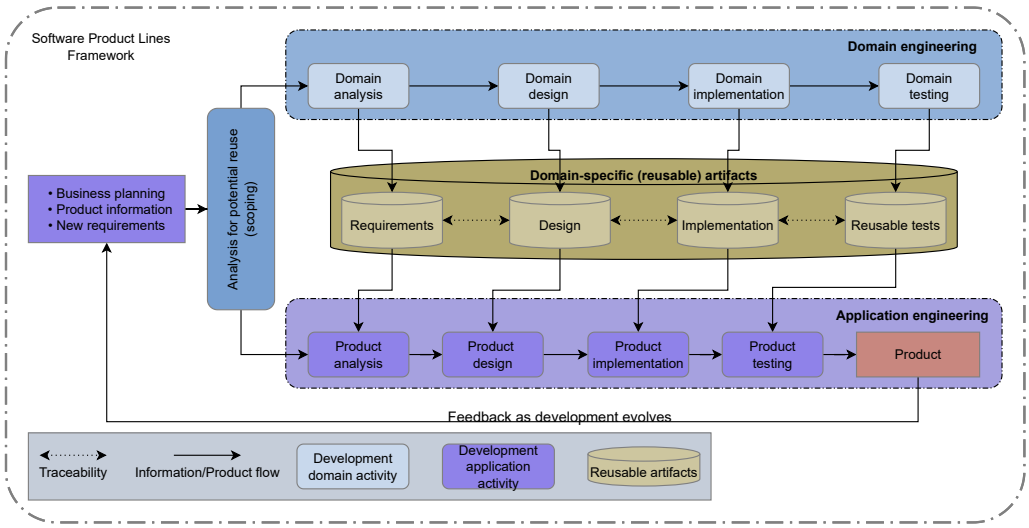

Figure 1. SPL framework, stages and interactions.

2.2. Variability

One of the most relevant aspects of SPLs is variability, which can be defined as the ability of an SPL to be subject to changes, customizations, configurations, or extensions for its particular use in a specific context, providing the SPLs framework with the necessary flexibility to achieve product diversification and differentiation [13]. We can introduce the variability in the SPL through the definition of reusable artifacts. These artifacts are included as part of the definition of a software family and, based on the inclusion or exclusion of features in the final product, allow for the creation of different products

from the generic set [12]. Several authors have proposed various models to manage the variability of an SPL at the domain engineering stage [14,15], most of them are based on the FODA analysis method [16].

We can find feature models (FM) based on the FODA method. An FM is a tree graph, where the root node represents the family of products, and the features of this family are grouped along the tree. These features can be assembled so that certain software products can be generated. The FMs have been a relevant topic in the study of SPL variability, showing the best evolution in terms of the number of articles and references published in the SPL [17]. Specific proposals on automatic construction and reasoning on FM can be found in [18,19].

2.3. Dynamic Software Product Lines

A static generative approach is generally used to develop an SPL, i.e., when attempting to reconfigure a given product built through an SPL, the product is required to be redeveloped or instantiated [20]. Nowadays, there are continuous changes in the requirements of a software product, which raises the need to adapt at runtime, expecting a certain degree of autonomy in the software to respond to the evolution of market conditions [21]. DSPLs extend existing approaches in today's product line engineering by moving their capabilities to runtime variability management, helping to ensure that system adaptations lead to desirable properties [2]. DSPLs are composed of two stages, the engineering stage, where the aim is to develop the execution environment, and the stage of managing the variability of the system in use [2]. Table 1, shows the differences between SPL and DSPL approaches.

Table 1. Differences between SPLs and DSPLs.

Software Product Lines	Dynamic Software Product Lines
SPLs are used in systems where variability behaves statically.	DSPLs are used in software systems where variability is constantly changing.
Variability management provides a description of the possible systems that can be produced.	Variability management provides the definition of various system adaptations at runtime.
Market segment identifies the common set of software products to be developed using the SPL approach.	Variation points specified identifies the level of reconfigurations supported by the DSPL.
The approach provides a framework for a set of individual software systems with common features.	The approach provides a unique system, which describes the basis for possible adaptations.
SPLs has two stages, domain and application engineering.	DSPLs have two life cycles, the engineering stage and runtime variability management.

2.4. Runtime Variability

Due to the difficulty anticipating all the variability required by SPLs, we can consider using DSPLs. The main challenge for DSPLs is runtime variability, which can be defined as the ability of software to adapt to fluctuations in the needs of the environment, user requirements and evolving resource constraints [4]. Variability is managed by creating runtime variation points [3]. Several approaches express changes in variability as a function of context [22] (e.g., in feature modeling [23]). Other approaches consider reconfiguration management based on machine learning using the MAPE-K control loop, which is an architectural approach to managing system variability through machine learning [24]. Table 2 shows the differences between runtime variability for SPLs and DSPLs.

Table 2. Differences between runtime variability for SPLs and DSPLs.

Variability in SPL	Runtime Variability in DSPL
Activation/deactivation of system features prior to deployment.	Activation/deactivation of system features after deployment.
Adding and removing features in SPL design engineering.	Adding and removing features at runtime.
To make a change to the system variability, the product must be instantiated from the design engineering phase in SPL.	Provide optimal reconfiguration of the possible points of variation of the system.

2.5. Self-Adaptive Systems

A self-adaptive system is a closed-loop system with a feedback loop intended to adjust to runtime changes [25]. To develop a self-adaptive system suitable for changes in software variability, understanding the nature of the system will help determine the process change and the factors affecting the change [26].

Weyns proposes two principles for determining what a self-adaptive system is [1]:

- External principle: A self-adaptive system can autonomously manage changes and uncertainties due to the requirements of the environment, the system itself and its objectives.
- Internal principle: A self-adaptive system contains two components for managing communication with and monitoring of the environment, as well as for executing changes in runtime variability.

As self-adaptive systems increasingly require autonomous behavior, runtime variability mechanisms must provide multiple binding options after implementation. The binding options can be satisfied using DSPLs. The mentioned conditions address the challenges of creating highly configurable software using runtime variability mechanisms to support automatic decision-making [27].

3. Related Work

Several secondary studies have been published on managing variability in SPL [17,28–30]. However, no secondary studies have been reported for variability management in DSPL for self-adaptive systems, obtaining only one systematic mapping study related to DSPL [9] that will be discussed below. Next, we present a summary of five proposals related to managing variability in SPLs or DSPLs [9,13,31–33]. Table 3 shows the overlapping RQs for related work (✓: fully answered, ∼: partially answered). For details on these related works (goals, RQs, and results), see the Appendix A.

Table 3. Overlapping RQs for related work.

Ref.	RQ1	RQ2	RQ3
[9]	∼		
[31]		∼	
[32]		✓	
[13]		∼	
[33]			✓

Guedes et al. [9] present an SMS on variability management in DSPLs, specifically on runtime variability modeling and configuration in DSPLs. The study was conducted between 2006 and 2015. The authors conclude that feature model are the primary approach to model variability in DSPLs, and most proposals do not refer to process configuration. Also, the authors declare that reconfiguration is triggered based on context, non-functional requirements, or user requirements through utility function or Constraint Satisfaction Problems (CSP).

Geraldi et al. [31] present an SMS for SPLs interacting with IoT technologies, seeking to identify approaches that manage the variability required by cyber-physical systems as their requirements evolve. This study analyzes the architectures used for IoT systems in which SPL manages variability between 2006 and 2018. The authors argued that several proposals seek to manage the variability of these systems in both runtime and SPL design, evolving towards a DSPL environment to manage variability through software reconfigurations.

Da Silva et al. [32] expose through a systematic literature review of the literature the most used practices in the product derivation stage in the DSPL approach. They identified several proposals, which base their reconfigurations on the system's architecture to be developed, from the software system's behavior, architecture, and components, which were mapped based on the MAPE-K control loop.

Chen and Ali Babar [13] present a systematic literature review in which they analyze various proposals to manage variability in SPL, obtaining that the feature models are the primary approach to manage variability in SPL, where in second place is the use of UML and its extensibility. Although this proposal does not focus on DSPL, it establishes an evolution of DSPL between the years 1990 to 2010, showing that the DSPL was not yet at the peak it is today.

Mohabbati et al. [33] present an SMS, looking for the main application areas of SPLs. This work is based on a service orientation from 2001 to 2011, describing that many of the software services created from SPLs are adaptive systems and, to a lesser extent, the DSPL approach. It presents a series of challenges for the integration of SPL and service-orientation, from theoretical models, taxonomies, or categorization of concepts, confirming that in SPL research, the goal is to achieve consistent and adequate modeling for service-oriented systems, supporting the configuration of dynamic and adaptive systems, reducing system complexity.

Finally, we can remark that our study shares a context with the previous articles. However, this SMS is explicitly oriented to dynamic variability management in DSPL in the context of self-adaptive systems. In addition, it considers approaches constraining runtime variability, methodologies used to manage the DSPL environment, and the main challenges.

4. Methodology

In order to obtain the status of DSPL variability management, an SMS was conducted according to the guidelines defined by Petersen [34]. In this section, we will describe the search and filtering process and then classify each item according to the research questions designed to study the dynamic variability management offered by DSPL. This protocol is based on a revised version published in arXiv [35], in which some sections were updated, such as the classification schema of RQs. Section 4.1 presents the definition of the protocol. Section 4.2 presents the process of the pilot selection of papers. Section 4.3 presents the classification scheme and data extraction from the papers. Section 4.4 presents the tools used during the elaboration of the SMS. Figure 2 presents a step by step summary of the methodology.

4.1. Protocol Definition

Next, we present the main phases to define the SMS protocol. The initial stage consists of determining the study's aim and need (Section 4.1.1). Then, the RQs and PQs associated with the fulfillment of the study objective will be defined (Sections 4.1.2 and 4.1.3). From the RQs, the search string (Section 4.1.5) will be generated to search for relevant articles in the data sources, in which the results will subsequently be filtered based on inclusion and exclusion criteria (Section 4.1.6). This process will end with validating the defined protocol (Section 4.1.7).

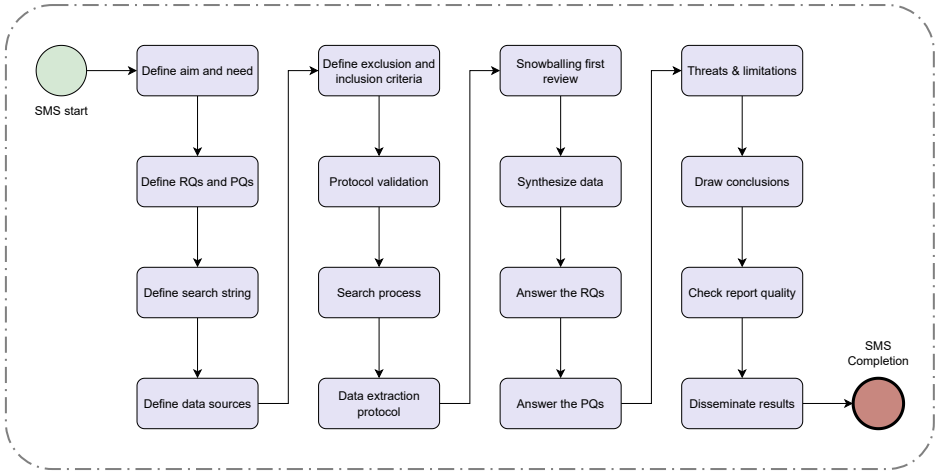

Figure 2. SMS process.

4.1.1. Aim and Need

The SMS aims to capture several approaches, methodologies, or design patterns for managing variability in DSPL, contextualized in self-adaptive systems. It is intended to map the most used practices to restrict and ensure the correct operation of software reconfigurations in DSPL. This mapping includes analyzing what types of approaches are used to manage variability at runtime, maintaining the stability and reliability of the system during the constant evolution of variability. Also, the SMS allows obtaining the main errors, difficulties, or challenges in developing DSPL and the process of system adaptation.

It is expected that the publication of the systematic mapping will contribute to identifying several challenges in managing variability for the DSPL scientific community, as well as contribute to the integral development of the DSPL through the generation of a line of research. This work is the initial part of a proposal that seeks to build and develop an architecture to manage variability in DSPL during the software reconfiguration process for self-adaptive systems, where it is necessary to know in detail the proposals that exist today in the area, as this will allow:

- Identify the requirements to generate a software reconfiguration while maintaining stability in the running system during the process.
- Know which technologies, tools, approaches, or others are used during the system reconfiguration process in DSPL and understand the justifications for use in each case.
- Avoid activities or processes already performed by other authors.
- Identify existing challenges in the area of reconfigurations for DSPL.

Moreover, the latest study collecting this information is seven years old [9], which mentions that DSPLs are still in a state of maturation; as far as variability runtime configuration is concerned, most proposals do not address process reconfiguration. This study will be contextualized in self-adaptive systems because this type of system correctly links the properties provided by DSPLs, delivering a high level of automation in managing runtime variability.

The importance of this study resides in systematically collecting and reporting an up-to-date view of the state of the art of leading approaches to manage runtime variability in DSPL, and this will be stored in terms of origin, type of approach to reconfiguring the system, the methodology used to implement the such approach and the latent challenges in DSPL. We have included other bibliometric aspects in this study, such as the article's origin, publisher, year of publication, and target group.

Providing an overview of all these features can help practitioners lower the risks associated with selecting an approach to managing variability in a self-adaptive system using DSPL. It can also contribute to the growth of the DSPL field by providing a framework in which different proposals for managing variability dynamically using this approach can be compared. It is intended to encourage discussion among the community on the practices

that should be deepened in the management of the DSPL, collecting various challenges proposed in the literature to mature the state of the art of DSPL.

4.1.2. Definition of Research Questions

The research will provide background on how variability is managed in DSPL's main challenges and constraints. The main question of our research refers to *What are the main difficulties of the approaches proposed in the literature to manage runtime variability during reconfiguration of dynamic software product lines in self-adaptive systems?* Then, it is necessary to know the existing proposals in the literature related to approaches for making changes in runtime variability. The generation of new information through this study will allow us to learn about the most relevant technologies and the context in which the respective approaches have been used. The general question was broken down into three questions related to the runtime variability management approach, the methodology used to manage the DSPL environment, and the main challenges encountered in the area. Table 4 shows the RQs, their aim, and a possible classification scheme.

Table 4. Research Questions, aim and classification scheme.

Research Questions	Aim and Classification Schema
RQ1. What approach was used to apply constraints during software reconfigurations in DSPL?	Collect information on the type of practices used to maintain system stability during DSPL reconfigurations: Closed Dynamic Variability, Open Dynamic Variability, Collaborative features, No proposed approach.
RQ2. What methodologies are currently used to manage DSPL variability during reconfigurations?	Identify how the variability model communicates with the running system to visualize reconfiguration changes: Proprietary architecture, MAPE-K, Agent-oriented software engineering, Third-party software, No specific methodology.
RQ3. What are the current challenges in the management of DSPL?	Highlight the main challenges to properly manage the dynamic variability of DSPL: Techniques, Open variation , Explicit variation points, Support of defaults, Binding time, Variant isolation, Proposal validation, Granularity, Non-code artifacts, No challenges mentioned.

4.1.3. Definition of Publication Questions

Additionally to the RQs, we included a set of publication questions (PQs) to complement the information collected and characterize the bibliographic and demographic space. These PQs include the type and place where papers are published and the number of papers per year. Table 5 show the details.

Table 5. Publication Questions and main goals.

Publication Questions	Aim and Classification Schema
PQ1. What year was the article published?	Highlight how DSPL research has evolved over the years. Years with more publications: 2010–2021
PQ2. Where was the article published?	Identify the journals and conferences most interested in the study of the DSPL, analyzing the most predominant of them and publishers.

4.1.4. Data Sources

According to [36,37] we consider the data sources detailed in Table 6, that are recognized among the most relevant in the Software Engineering community.

Table 6. Publication Questions and main (Accessed Thursday 5th May 2022).

Library	URL
ACM Digital Library	dl.acm.org
IEEE Xplore	ieeexplore.ieee.org
Science Direct	sciencedirect.com
Springer Link	springer.com
Wiley Inter-Science	onlinelibrary.wiley.com

4.1.5. Search Strategy

The search strategy begins with elaborating the search string, which has been defined according to the steps defined by [36]. Through the context and RQs, a set of keywords has been extracted. Then, a set of synonyms has been proposed for each keyword to widen the search range. Using PICOC (*Population—Intervention—Comparison—Outcomes—Context*) [38], the search string for the systematic mapping is constructed.

- Population: In the Software Engineering community, the population includes a specific role, category, application area, or industry group. We selected an application area, specifically runtime variability management in DSPLs.
- Intervention: An intervention consider a methodology, procedure, technology, or tool that addresses a specific issue. We selected a procedure, specifically the proposals for runtime variability management.
- Comparison: We do not consider comparing the selected papers against a specific variability management proposal (control condition). Then, these criteria do not apply to our study.
- Outcomes: The outcomes of our RQs are the approach to managing variability and the architecture used for the problem solved in each proposal.
- Context: The context for this study are DSPL, reconfiguration proposals to manage runtime variability, and self-adaptive systems.

The list of keywords and synonyms is defined as follows:

- *adaptation, reconfiguration*
- *dynamic software product lines, software product lines*
- *self-adaptive systems, adaptive systems, evolution*

It was decided to add the term "evolution" to the keywords because it is mentioned by several authors to refer to the evolution of SPLs towards DSPLs [3,39,40]. The search string changed its structure since, in the beginning, it contained terms associated with the classification of research questions, such as *"framework", "methods", "tools", "classification", "architecture", "restrictions"*. It was decided to exclude these terms since they did not influence the number of articles found in the different data sources. The final query string is described as follows:

("self-adaptive systems" OR "self-adapting systems" OR "auto-adaptive systems" OR "self-adaptive" OR "self-adapting" OR "adaptive systems" OR "self-adaptation" OR "adaptation")
AND
("dynamic software product line" OR "dynamic software product lines" OR "dynamic product family" OR "dynamic product families" OR "dynamic product line" OR "dynamic product lines" OR "Software product line" OR "software product lines" OR "product family" OR "product families" OR "product line" OR "product lines")
AND
("system reconfiguration" OR "reconfiguration" OR "reconfiguration rules" OR "configuration rules" OR "adaptation rules" OR "evolution")

It is essential to mention that due to the limitations of some search engines, we have to adapt the search [41]. However, these adaptations do not add or remove any filters. Table 7 shows the search string results used in the selected data sources. Due to the restriction in

the search string of the *Science Direct* data source, where a maximum of eight key terms is allowed, the following search string is used:

("self-adaptive systems" OR "self-adaptive" OR "auto-adaptive systems" OR "adaptation") AND ("dynamic software product line" OR "dynamic software product lines" OR "software product lines" OR "software product line")

Table 7. Search string, total results (Accessed Thursday 5th May 2022).

Library	Result
ACM Digital Library	765
IEEE Xplore	74
Science Direct	485
Springer Link	5511
Wiley Inter-Science	1011
Total Result	7846

4.1.6. Selection Process

The search process begins with an automatic search using the selected data sources and the search string defined in Section 4.1.5. The goal is to obtain the first collection of articles to be distributed among the team's researchers. Once the first collection of articles has been obtained, each researcher must independently filter each article according to the following inclusion/exclusion criteria, which allow us to determine whether an article is relevant to our study or not. This process is determined by reviewing only each article's title, abstract, and keywords. Inclusion criteria (IC) and exclusion criteria (EC) will be applied to each selected article. These criteria are shown in Tables 8 and 9. To avoid excluding relevant articles, we will occupy a first snowballing search review according to the guidelines proposed by Wohlin [42]. The search range will be extended by reviewing for every selected article the references of each article (backward snowballing) and the citations obtained by the articles (forward snowballing).

Table 8. Inclusion criteria.

ID	Criteria
IC1	Articles published between 2010–2021.
IC2	Papers written in English.
IC3	Type of paper: • Proceeding. • Journal. • Conference Paper. • Chapter LNCS (Lecture Notes in Computer Science).
IC4	Papers with more than one version, only the latest version will be included.
IC5	Papers whose abstracts deal with Dynamic Software Product Lines or reconfigurations in Software Product Lines for self-adaptive systems.
IC6	Topic: • Computer Science.

4.1.7. Protocol Validation

Validation was performed by defining and reviewing each step of the protocol. This review was carried out by one of the authors, who has extensive experience conducting secondary studies. An initial version of this protocol was published on the arXiv platform [35]. The protocol presented in this paper corresponds to each step's final result (definition plus validation). During the search process, weekly meetings were held to discuss doubts in selecting papers applying inclusion and exclusion criteria. These meetings were held through the Zoom platform.

Table 9. Exclusion criteria.

ID	Criteria
EC1	Articles written before 2010.
EC2	Articles not related to Software Product Lines.
EC3	Secondary researches (If they exist and are relevant to the research, they will be added as related work).
EC4	Papers without access.
EC5	Duplicate papers will be excluded.
EC6	The following types of items will be excluded: • Tutorials. • Short papers with less than four pages. • Abstract. • Poster. • Paper in progress (incomplete). • Book and book chapter.

4.2. Pilot Selection

Before the selection process and subsequent classification of papers, we conducted a pilot screening and extraction process among the investigators to ensure the reliability of the protocol.

To avoid any possible bias due to each investigator examining a different set of articles, we confirmed that the application of the IC and EC was similar between the two investigators and the assistant (inter-rater agreement). This check was carried out individually by each team member, verifying compliance with each IC and EC over a set of 12 papers chosen randomly from among those recovered in this pilot selection process.

As a means of verification, we performed a concordance test based on *Fleiss' Kappa statistic* [43]. We obtained a *Kappa = 0,81* in the concordance check between the team members. This value suggests that the criteria were clear enough for the research team to apply the IC and EC consistently [44].

4.3. Classification Scheme and Extraction

We seek to compile a complete list of articles related to DSPL proposals, where there is a component dedicated to managing runtime variability. This SMS will seek to map articles between the years 2010 to 2021, since according to a study conducted by Guedes [9], between the years 2006 to 2010, the DSPL area was not booming, containing a minimal amount of articles in those years. We searched from January to July 2022. Once the protocol was validated, we began the phase of retrieval of primary studies and data extraction. The first step to perform the extraction corresponds to the execution of the search string in the data sources, where we obtained 7.846 papers, which can be seen in Table 7. At this stage, a decrease in the number of papers in the IEEE Xplore data source can be visualized due to the second segment of the search string related to software reconfigurations.

Next, the papers were filtered in three stages, according to the priority defined in Figure 3. The first stage consisted of direct filtering of the data sources, such as year and topic, resulting in 1.829 papers. The second stage consisted of the execution of basic filters, such as the verification of the type of article, the articles relevant to the research topic, and the exclusion of secondary studies, resulting in 352 papers. The third filter was based on excluding duplicate papers and articles focused on the software reconfiguration process using DSPL, obtaining a final result of 68 relevant papers.

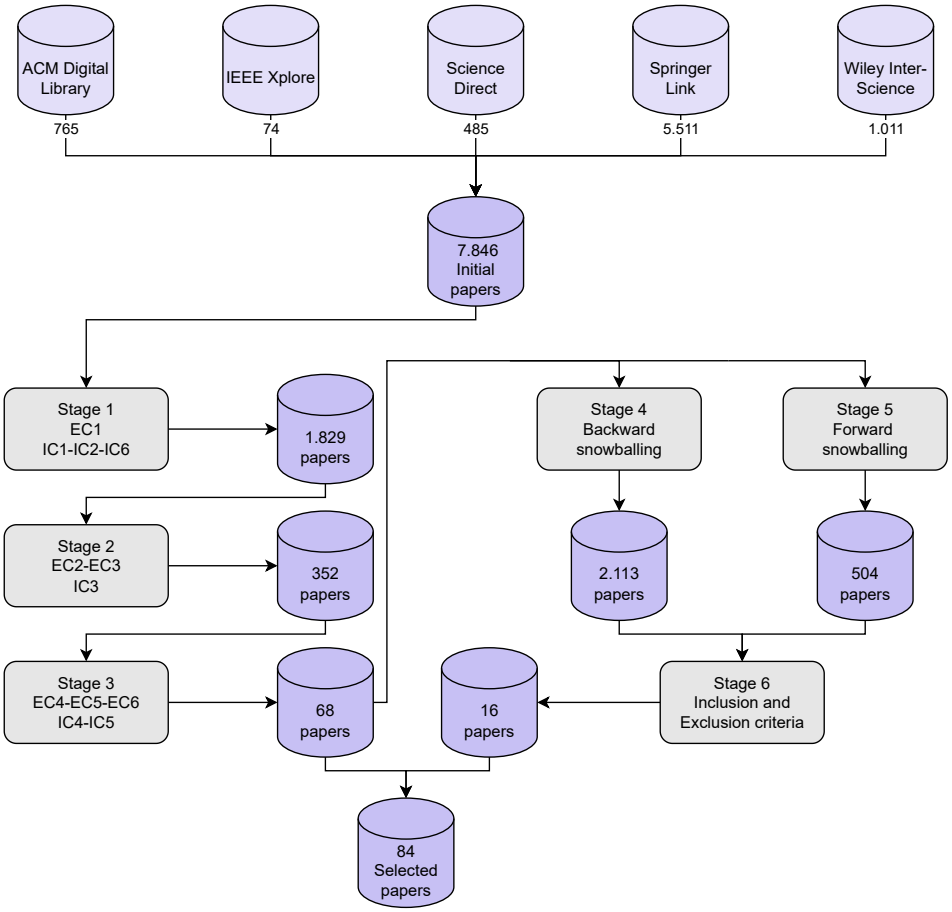

Figure 3. SMS Selection process.

After obtaining the relevant articles from the data sources, a complimentary search was carried out through snowballing. This process reviews the citations and references of the first selection of articles. In the references of each article (*backward snowballing*), a total of 2.113 papers were obtained. On the other hand, in the citations of each article (*forward snowballing*), a total of 504 papers were obtained. In the case of the forward snowballing search, the Scopus data source was used as support (scopus.com, accessed 1 June 2022). This support was considered because there were some papers with inconsistencies between the number of citations concerning what was reflected by their data source (e.g., the proposal of Rosenmuller et al. was cited by 32 papers [22], but the data source provides only eight papers). We applied the same IC and EC to the papers obtained through the snowballing process, resulting in 16 relevant papers. Then, we obtained 84 papers for the SMS. For details of selected papers, see Appendix B.

The articles that meet the IC and EC mentioned in Section 4.1.6 will be stored in a Google spreadsheet in which the following information from the metadata collected for each article will be entered:

- Title.
- Authors (each one).
- Year of publication.
- Publication type and classification (conference, journal).
- Approach used to manage runtime variability in DSPL.
- methodologies used to manage variability.
- Challenges in DSPL management.
- Results and future work.

4.4. SMS Tool Support

We used the following support tools to facilitate online collaboration among team members. Google Drive (drive.google.com) and Google Sheets (docs.google.com/spreadsheets) were used to search, document, analyze, filter and classification of papers. Google Scholar (scholar.google.com) was used for testing and validating the search string. Overleaf (overleaf.com) was used to edit and manage the article writing process. Based on the conditions imposed by the Coronavirus pandemic, the team used Slack (slack.com) and Zoom (zoom.us) tools to communicate and coordinate the work. We use another set of tools for bibliometric analysis. To create the word cloud tag, we used TagCrowd (tagcrowd.com), and for the Sankey diagram, we used Sankeymatic (sankeymatic.com). To find relationships between the collected data and for subsequent visualization in figures, we used TerMine (TerMine) and VOSviewer (vosviewer.com). Cabuplot (cabuplot.herokuapp.com) was used to generate the bubble chart [45].

5. Results

This section reports the results to the RQs (Section 5.1) and PQs (Section 5.2) raised by this SMS.

5.1. Answers to RQs

In the following, we report the results for RQ1 (Section 5.1.1), in which we will report the main approaches used to manage runtime variability. For the results for RQ2, which seeks to obtain the primary methodologies to manage DSPLs, see Section 5.1.2. Answering RQ3 is based on finding the main challenges in managing DSPLs see Section 5.1.3.

5.1.1. RQ1: What Approach Was Used To Apply Constraints during Software Reconfigurations in DSPL?

We check the constraints on runtime variability to learn about the most commonly used approaches and their context. When classifying the papers, a significant problem arose related to the initial classification of this RQ presented in the SMS protocol [35], since there were dimensions with different granularity (some very specific and others general). Therefore, to answer this RQ, the classification was based on the guidelines by Mens et al. [46], in which the authors present a taxonomy of software variability approaches.

- Closed Dynamic Variability: This approach aims to support the dynamic activation and deactivation of features that have been predefined in advance in the runtime variability design. A potential execution scenario can be a smart home system, where the context data collected by the installed sensors activate or deactivate some predefined features within the system, allowing it to be implemented in any smart system using sensors. In order to apply new variants to the system would require redesigning the feature model, as this approach does not support unforeseen scenarios, contexts, or features.
- Open Dynamic Variability: This approach allows for unforeseen changes in the design variability at runtime, supporting the addition, removal, and modification of features dynamically, facilitating its application in systems that may need to cope with unforeseen scenarios (e.g., smart cities, robots). This variability mechanism supports unforeseen scenarios in a controlled manner.
- Collaborative features: Seeks to represent features that can exchange context information at runtime for specific purposes, e.g., for real-time and critical systems or swarm systems. The main limitation of this approach is that dynamic changes are not supported.
- No proposed approach.

Forty-eight papers (57.1%) present an approach based on *Open Dynamic Variability*. Thirty-two papers (38.1%) present an approach based on *Closed Dynamic Variability*. Three papers (3.6%) present a *Collaborative features* approach and one paper (1.2%) *No proposed*

approach. See details in Figure 4 and the classification of each item according to the criteria previously defined in Table 10.

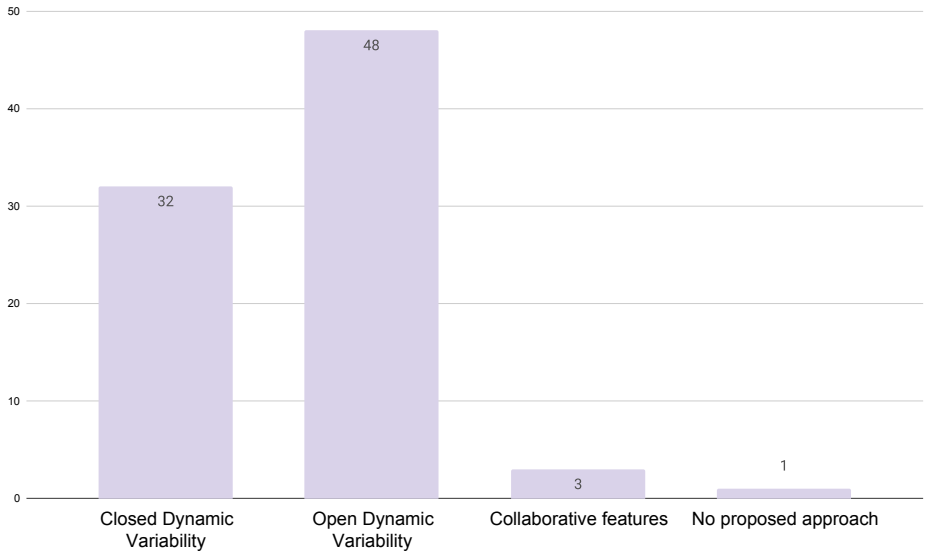

Figure 4. Approaches to managing runtime variability in DSPL.

Table 10. Selected papers and approaches to managing runtime variability in DSPL.

Approaches for Applying Runtime Variability Constraints in DSPL	Selected Papers
Closed Dynamic Variability	SP1, SP4, SP5, SP6, SP7, SP10, SP11, SP12, SP13, SP14, SP15, SP17, SP18, SP19, SP20, SP21, SP24, SP27, SP29, SP32, SP37, SP44, SP46, SP49, SP55, SP59, SP62, SP68, SP69, SP70, SP73, SP78.
Open Dynamic Variability	SP2, SP3, SP8, SP22, SP23, SP25, SP26, SP28, SP30, SP31, SP33, SP34, SP36, SP38, SP39, SP40, SP41, SP42, SP43, SP45, SP47, SP48, SP50, SP51, SP52, SP53, SP54, SP56, SP57, SP58, SP60, SP61, SP63, SP64, SP65, SP67, SP71, SP72, SP74, SP75, SP76, SP77, SP79, SP80, SP81, SP82, SP83, SP84.
Collaborative features	SP9, SP16, SP35.
No proposed approach	SP66.

5.1.2. RQ2: What Methodologies Are Currently Used to Manage DSPL Variability during Reconfigurations?

RQ2 seeks to obtain further indications on the most used methodologies in the management of DSPLs and their runtime variability, mainly visualizing their application in self-adaptive systems. This research question will be classified according to five dimensions, which were grouped once each article was analyzed.

- Proprietary architecture: Methodologies proposed by the authors are mainly composed of context-aware feature models. The variability in execution time is specified during the design stage. It also contains proposals for reconfigurations based on optimization models or constraint language in variability modeling.
- MAPE-K control loop: Control loop for self-adaptive systems presented by Kephart and Chess [6], which allows managing runtime variability through machine learning.

This loop consists of five stages [7]. The stages of MAPE-K control loop are presented in Figure 5.

1. Monitor: Module in charge of monitoring managed resources and collecting, grouping and filtering data. Monitoring is done through sensors, e.g. IoT devices.
2. Analyze: Module responsible for analyzing the data provided by the monitor, understanding what the current status is and whether measures should be taken to mitigate needs.
3. Plan: Module responsible for developing an action plan based on the results of the analysis, this being a set of measures that will take the system from its current state to the desired state.
4. Execute: Module responsible for the execution of the action plan and the follow-up of the measures adopted in the managed element.
5. Knowledge: Knowledge is the central node of the control loop and is accessible to all components of the loop, incorporating in addition to the data collected and analyzed, additional elements such as architectural models (in the DSPL context, feature or goal models), policies and change plans [6].

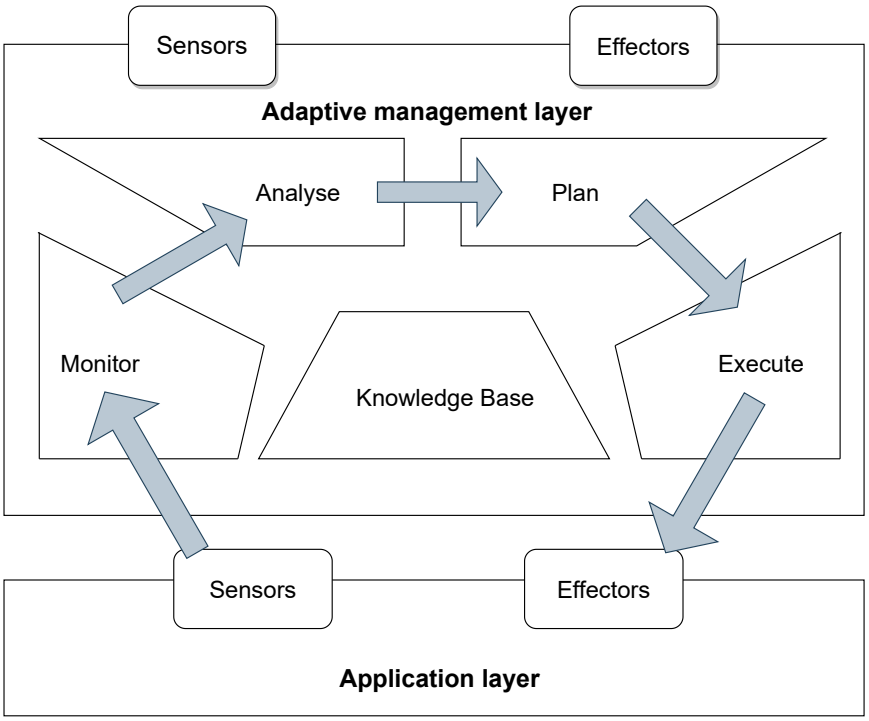

Figure 5. MAPE-K control loop.

- Third-party software: DSPL proposals that require the use of external software for their main operation.
- Agent-Oriented Software Engineering: Proposals in which methodologies associated with software agents are applied, such as multi-Agent Systems Product Lines.
- No specific methodology.

Fifty-one (60.7%) papers present a *Proprietary architecture* for managing a DSPL execution environment. Twenty-one (25%) papers present the *MAPE-K control loop* as a runtime reconfiguration engine. Seven papers (8.3%) present an architecture based on *Agent-Oriented Software Engineering* to manage DSPL environments. Four papers (4.8%) use *Third-party software* to manage a DSPL environment and one paper (1.2%) presents *No specific methodology*. See details in Figure 6 and the classification of each item according to the criteria previously defined in Table 11.

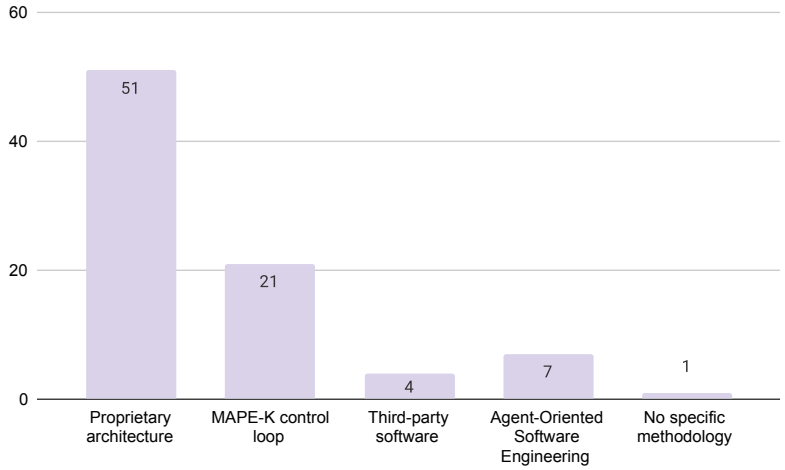

Figure 6. Methodologies for managing DSPLs.

Table 11. Selected papers and methodologies for managing DSPLs.

Methodologies Used to Manage DSPL Runtime Variability	Selected Papers
Proprietary architecture	SP1, SP4, SP5, SP6, SP7, SP9, SP10, SP11, SP13, SP15, SP17, SP18, SP19, SP20, SP21, SP24, SP26, SP27, SP29, SP31, SP32, SP35, SP37, SP41, SP43, SP44, SP45, SP46, SP48, SP49, SP52, SP54, SP55, SP59, SP60, SP62, SP63, SP64, SP65, SP66, SP67, SP68, SP70, SP71, SP72, SP73, SP74 SP78, SP81, SP83, SP84.
MAPE-K control loop	SP2, SP3, SP8, SP23, SP28, SP30, SP33, SP34, SP36, SP39, SP40, SP42, SP50, SP57, SP58, SP61, SP75, SP77, SP79, SP80, SP82.
Third-party software	SP12, SP14, SP25, SP69.
Agent-Oriented Software Engineering	SP16, SP38, SP47, SP51, SP53, SP56, SP76.
No specific methodology	SP22.

5.1.3. RQ3: What Are the Current Challenges in the Management of DSPL?

Finding the most relevant challenges will serve to develop a continuous improvement in the development of the DSPL and satisfy the main problems mentioned by the authors. Similar to question RQ1, a change was made concerning the initial classification in the SMS protocol since it was not known what challenges precisely existed in managing the DSPL. Therefore, to answer this research question, the classification will be based on the guidelines proposed by Zhang et al. [47], which proposes areas of application of the variability mechanisms. Additionally, a section related to the validations of the proposal, either in different application areas or in industrial systems, was added.

- Techniques: While some mechanisms use general techniques supported by almost all programming languages (e.g., cloning, conditional, and module execution), other mechanisms can only be applied in programming languages or even require a specific environment or tool (Aspect Aspect Orientation, Frame Technology).
- Open variation: Mechanisms that allow open variability can enable external developers to provide software artifacts that extend the system after compilation while allowing the kernel to be treated as a "black box" with additionally defined points of variation.
- Explicit variation points: These are mechanisms in which the possible changes it may contain in its variability are explicitly defined.

- Support of defaults: Seeks to support and mitigate possible errors that runtime variability can generate. For example, Zhang mentions that in some cases where the default selection at a variation point can reduce the number of variants (by one) and simplify the variation logic [47].
- Binding time: In the software lifecycle, specific features are instantiated and bound to a variant at a specific time, either at build time (preprocessing and compilation) or run time. Maintaining a mechanism to manage binding times allows variability to be configured in advance and reconfigurations to be optimized.
- Variant isolation: The code variants for each point of variation are written in a source file or isolated modules or files, allowing grouping, for example, of backup systems.
- Proposal validation: Proposed frameworks to manage variability in execution time, where the methodology must be validated in several areas because they are laboratory tests.
- Granularity: Depending on the variability mechanism used, the granularity of variants differs. Some mechanisms are based on textually describing variants or variation points, admitting any granularity in a source file. In contrast, other mechanisms impose a specific size or shape of variants within the code structure (e.g., a function, a class, or a file).
- Non-code artifacts: Software systems include several types of artifacts in addition to code, for example, variability modeling, data files, or text files. However, some of the mechanisms presented are applicable only to code.
- No challenges mentioned.

Twenty-four (28.6%) papers present challenges associated with *Techniques*. Eighteen papers (21.4%) present challenges in the area of *Open variation*. Fifteen papers (17.9%) present as a result the *validation of the proposal*. Four papers (4.8%) present challenges in the area of *Explicit variation points*. Three papers (3.6%) present challenges with *binding time* in DSPLs. Two papers (2.4%) show challenge the *Variant isolation*. One paper (1.2%) has challenges with *Non-code artifacts*. One paper (1.2%) contains challenges related to feature *Granularity*. One paper (1.2%) presents challenges in the area of *Support of defaults* and fifteen papers (17.9%) have *No challenges mentioned*. See details in Figure 7 and the classification of each item according to the criteria previously defined in Table 12.

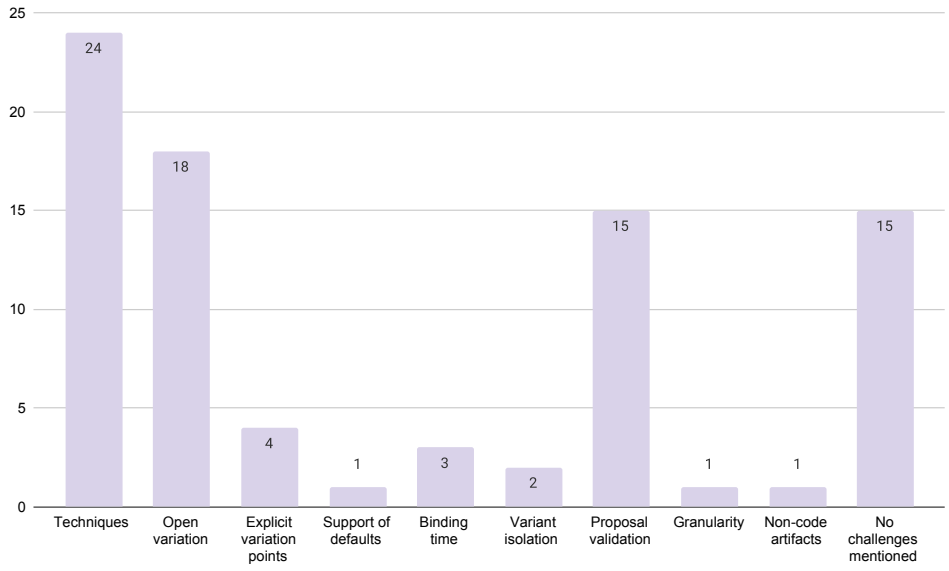

Figure 7. Challenges in managing DSPLs.

Table 12. Selected papers and DSPL management challenges.

DSPL Management Challenges	Selected Papers
Techniques	SP1, SP3, SP4, SP5, SP6, SP7, SP9, SP10, SP12, SP15, SP19, SP20, SP22, SP25, SP27, SP42, SP46, SP50, SP53, SP63, SP64, SP65, SP70, SP78.
Open variation	SP2, SP8, SP23, SP26, SP30, SP32, SP35, SP36, SP38, SP40, SP54, SP57, SP66, SP67, SP68, SP72, SP80, SP81.
Explicit variation points	SP11, SP14, SP24, SP39.
Support of defaults	SP13.
Binding time	SP16, SP18, SP49.
Variant isolation	SP21, SP59.
Proposal validation	SP34, SP43, SP44, SP45, SP48, SP55, SP60, SP61, SP69, SP74, SP75, SP76, SP79, SP82, SP84.
Granularity	SP37.
Non-code artifacts	SP58.
No challenges mentioned	SP17, SP28, SP29, SP31, SP33, SP41, SP47, SP51, SP52, SP56, SP62, SP71, SP73, SP77, SP83.

5.2. Answers to PQs

5.2.1. PQ1: What Year Was the Article Published?

To highlight the evolution of DSPL-oriented proposals in recent times, we identified the year in which each paper was published, in this case, between the years 2010 and 2021. See Figure 8 for details. For papers published each year, see Figure 9.

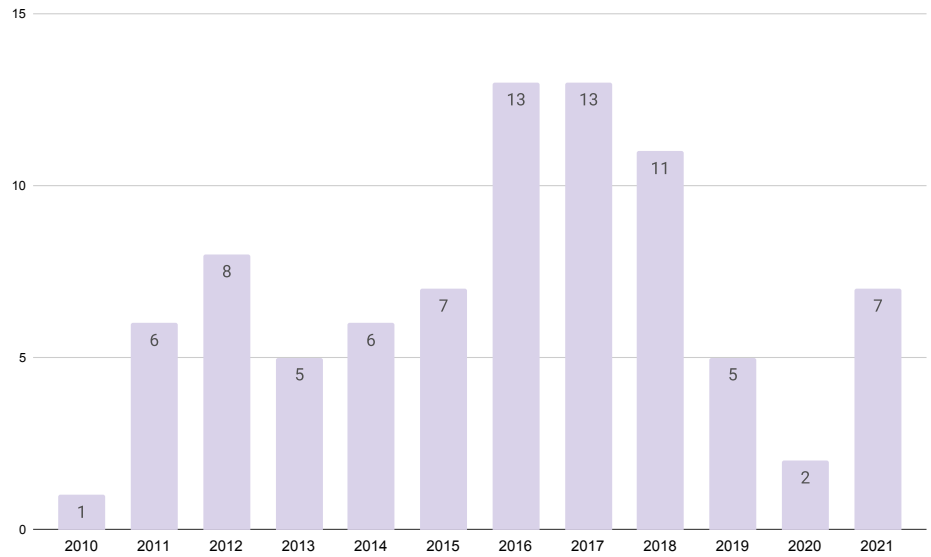

Figure 8. Number of papers published by year.

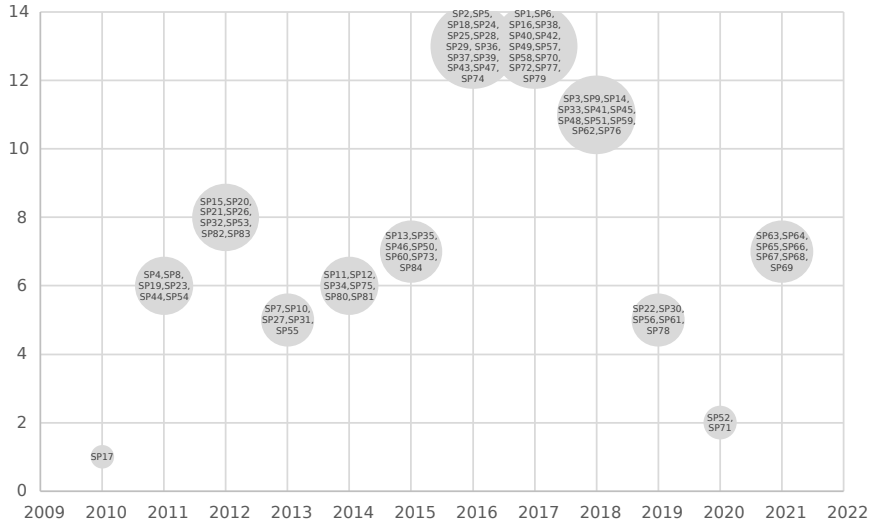

Figure 9. Selected papers published by year.

5.2.2. PQ2: Where Was the Article Published?

To assist researchers in identifying where there are more papers in the area of DSPLs, we classified the journals or conferences that were most interested in our area of study by checking where the papers were published. The ranking of the publication question considers the Conference and Journal dimensions. Fifty-six (66.7%) papers belong to Conferences, and twenty-eight (33,3%) papers were published in Journals. The detail can be seen in Figure 10.

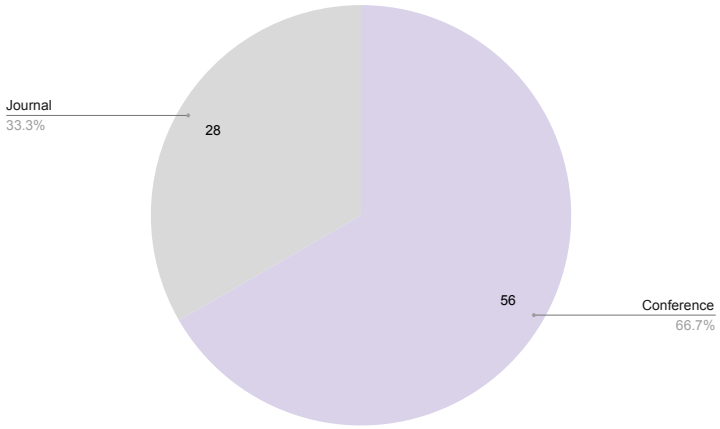

Figure 10. Number of articles according to the sources where they are published.

Moreover, we identify the publisher of each article. Thirty-five (41.7%) papers are published in ACM. Although few articles were obtained in the search performed in the IEEE data source, twenty-one papers (25%) were obtained, validating the search string. Fifteen papers (17.9%) were published in Science Direct. Nine papers (10.7%) were published in Springer, and two papers (2.4%) were published in Wiley. Due to the snowballing process, two more dimensions were obtained, obtaining one paper (1.2%) in World Scientific and MDPI, respectively. Details can be seen in Figure 11.

Also, we identified the journal or conference in which each article was published. For articles published in journals, see Figure 12; for those published in conferences, see Figure 13. The definition of each acronym will remain in Appendix C.

Next, we present a thorough analysis of the journals and conferences. First, in the case of the journals, twenty-eight papers came from eighteen journals, which were classified according to their indexing (Wos or Scopus). Figure 14 shows the distribution of the journals

according to their indexing. Figure 15 shows the distribution of the journals indexed in WoS JCR quartiles.

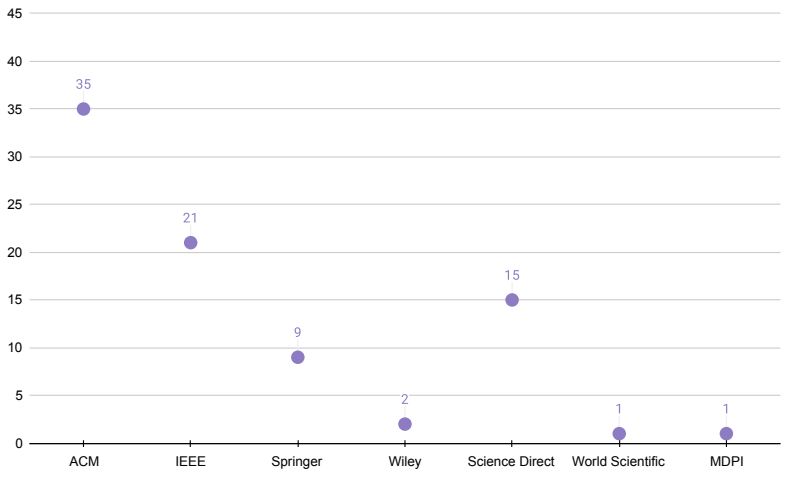

Figure 11. Number of papers according to publisher.

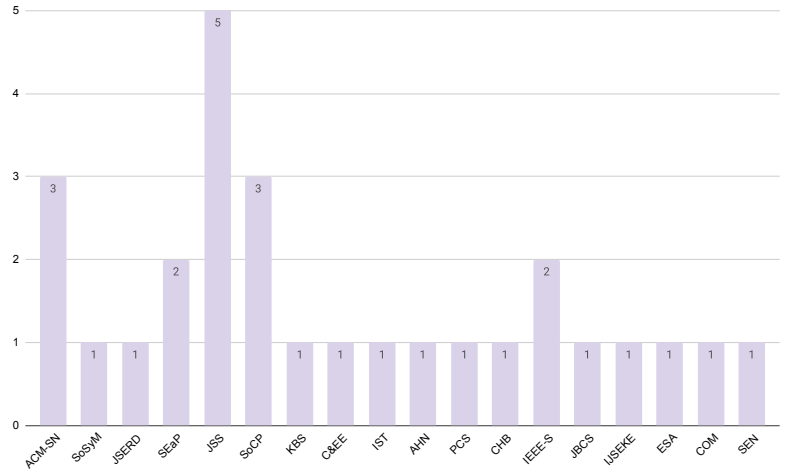

Figure 12. Papers published in journals.

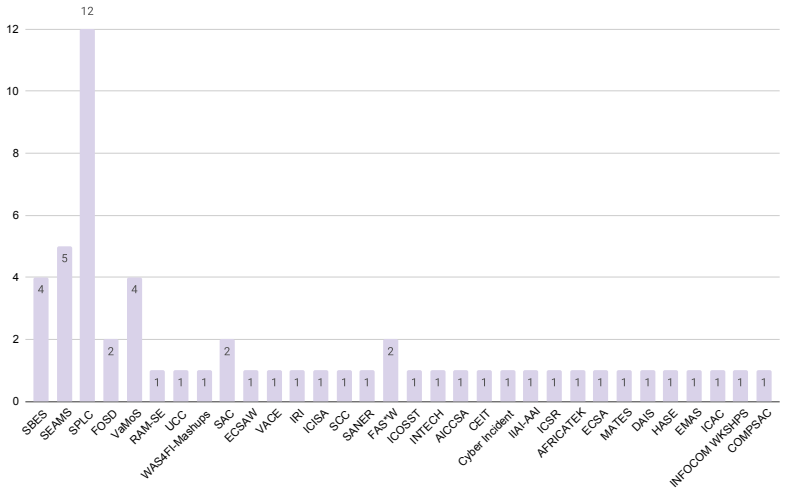

Figure 13. Papers published in conferences.

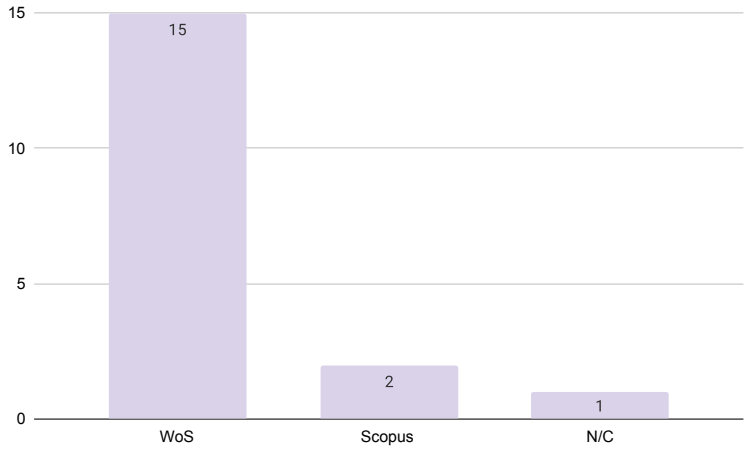

Figure 14. Journals according to their indexing.

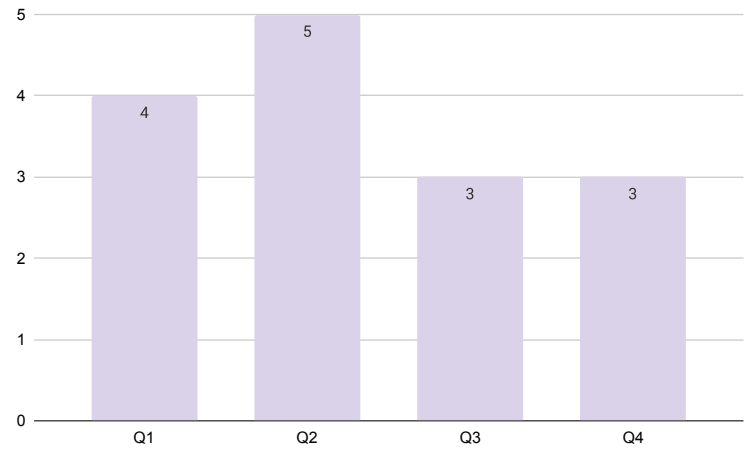

Figure 15. Journals according to WoS JCR quartil indexation.

Secondly, in the case of the conferences, fifty-six papers came from thirty-two conferences, which were classified according to their classification (ranking Qualis). Figure 16 shows the distribution of the papers, according to the Qualis ranking.

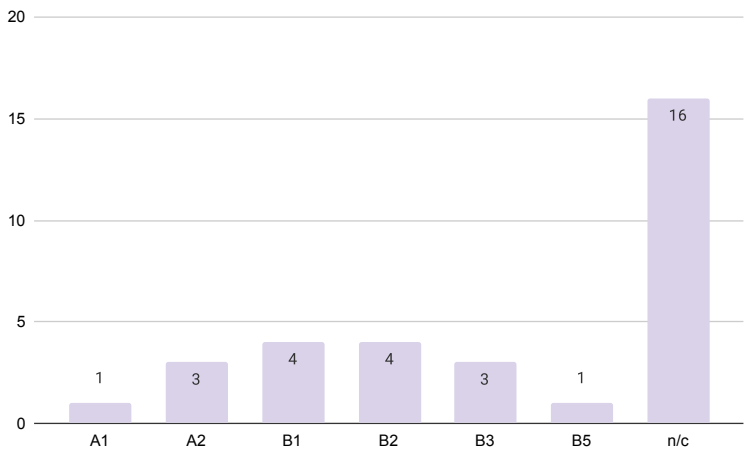

Figure 16. Conferences according to their indexing in Qualis ranking.

6. Discussion

This section analyzes the previous results. Section 6.1 provides an interpretation to handle the responses to the RQs and PQs. Section 6.2 analyzes the relationships between the RQs and the identified categories. A bibliometric analysis is presented in Section 6.3. Finally, Section 6.4 debates the main threats to the validity of this SMS.

6.1. Interpreting Answers to RQs and PQs

According to [3], the methodologies for managing runtime variability are in line with our classification, within which there are several challenges and execution scenarios for DSPL environments. Each of the RQs and PQs are discussed in detail in Sections 6.1.1 and 6.1.2 below.

6.1.1. Interpreting Answers to RQs

Interpreting Answers for RQ1

According to the collected evidence for answering RQ1, the two main approaches to managing reconfigurations are: (i) closed dynamic variability (38%) and (ii) open dynamic variability (57%).

Most authors present constraints on modeling variability for closed dynamic variability in either context-aware feature or objective models. Both seek to model possible adaptation rules for the system [24,48]. Also, there are proposals related to optimization models, for example, a prediction module to plan system adaptations [49], and reasoning engines [50], such as a SAT solver to check the satisfaction of the required reconfiguration concerning the feature model, seeking to analyze if any rule integrated into the model does not break.

In the case of open dynamic variability, there are several proposals; where most of them are based on learning future reconfigurations through machine learning [7,39,40,51–61], or some variant through software agents, generating Multi-Agent Systems Product Lines (MAS-PL) environments to manage the system through autonomous agents [62,63]. There are proposals associated with modeling constraints, such as [64], which proposes an approach to unify design and runtime adaptation based on aspect-oriented modeling through a metamodel of unified aspects of a platform that performs processes to achieve design and runtime adaptations. The approach associated with collaborative features, proposes a family-based analysis that simulates all product variants of a DSPL simultaneously, at runtime, on recent environmental inputs to obtain an estimate of the quality of service that each variant of the product would have had if it had been run at runtimeref [65].

Finally, the Figure 17 shows an increase in approaches based on open dynamic variability. Also, we can observe a decrease in proposals based on closed dynamic variability. This decrease could be due to the limitations generated by limiting the number of reconfigurations for self-adaptive systems.

Interpreting Answers for RQ2

According to the evidence provided by RQ2, most authors use their own methodology to manage a DSPL environment (61%). Most of these proposals are based on constraints mentioned in the variability modeling, using context-aware feature models, optimization models, or adaptation rules, among others. For example, the DAMASCo proposal seeks to provide reconfiguration to support specific runtime failure situations, grouping services into families that facilitate the selection and use of similar services in case of failures [66]. There are also DevOps-based architectures for reconfiguring self-adaptive systems [67].

The second majority of the work is in the MAPE-K control loop (25%). Shen presents variations of the control loop, including a metamodel to express the variability between the SPL and the self-adaptive system [68]. An instantiated model includes the business requirements, context, and adaptation logic. Nascimiento designs the variability of SPLs with the common variability language (CVL), which communicates with an architectural tool diagram similar to the UML component diagram, which communicates with the

MAPE-K control loop for reconfiguration [69]. Most articles in the Agent-Oriented Software Engineering category worked under the MAS-PL methodology to manage variability.

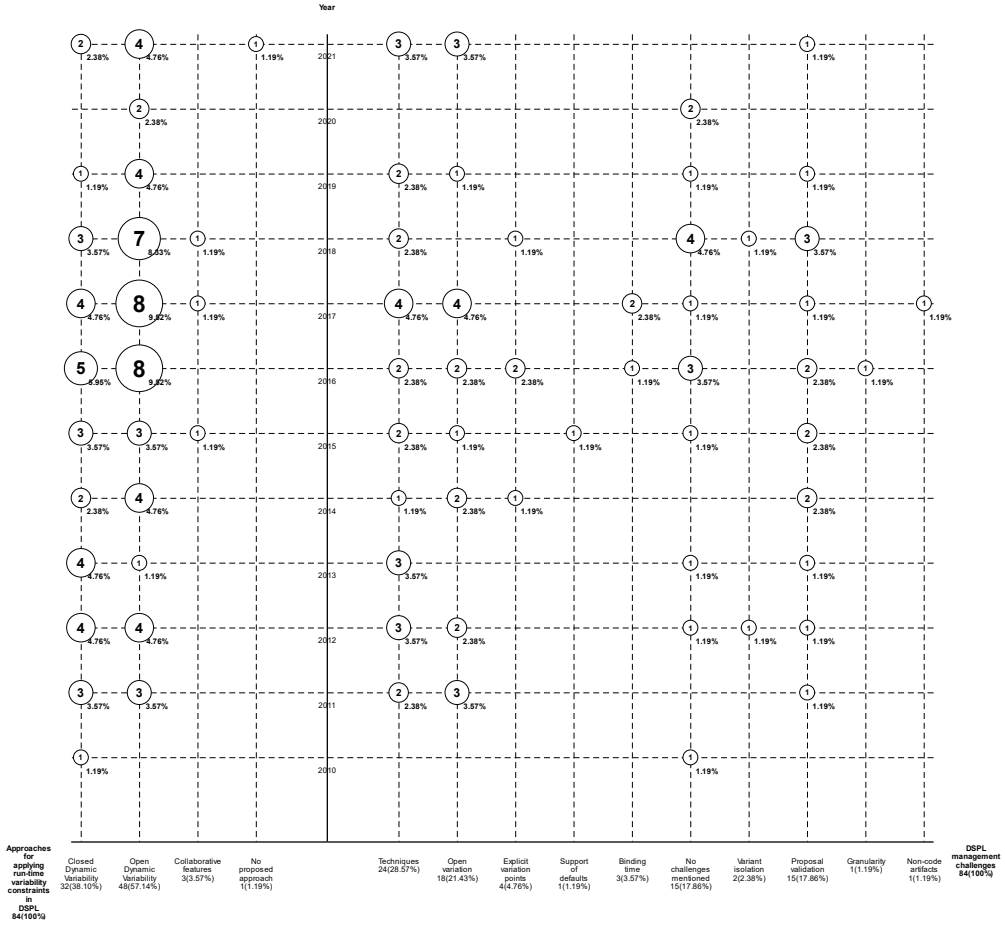

Figure 17. Relationship between approaches to managing runtime variability and the evolution of challenges to manage DSPLs between the years 2010 to 2021.

The above classification identifies two main approaches for managing variability: the MAPE-K control loop and FMs.

As mentioned in Section 5.1.2, the MAPE-K control loop corresponds to an autonomous control loop. Its main objective is to propose a framework to monitor, analyze, plan and execute reconfigurations in the software, storing the log of each step in a knowledge node, promoting the linkage with Machine Learning techniques.

Runtime variability management techniques associated with FMs have several variations, with several approaches having the same goal of managing the runtime variability of a system. Most of these systems store within the models specific variation points that the software can assume. For example, FCORE, a model-based approach, facilitates the coordination of adaptations between variability-intensive systems [70]. The allowable runtime reconfigurations of each system are specified using an FCORE model, combining the FMs used in the DSPL with the goal models. Then, the previous models are converted to constraint satisfaction problems for determining conflicts and synergies between system adaptations during execution. Lochau et al. [71] propose an approach based on extended feature models (EFMs) for simple FMs with feature attributes, binding times, and corresponding constraints. All model transformations have been implemented using eMoflon, a meta-CASE tool for model-based software development.

Among the most used approaches based on FMs to manage software reconfigurations are the Context-aware Feature Models (CFMs). This approach contains two modeling stages, the FM and the context model, which allows for providing an autonomous (pre) planning

and execution of the reconfiguration of a product at runtime according to contextual requirements identified for the DSPL [72]. An extended context-aware feature modeling (eCFM) is proposed by [73], which can identify common design flaws in DSPL FMs by identifying how these features behave over time.

Although the MAPE-K control loop and CFMs are two independent approaches, combined proposals address both CFMs and the MAPE-K control loop [74]. Table 13 shows a contrast between the proposed approaches with FMs and the MAPE-K control loop. In Figure 17, we can observe that the principal methodology applied to manage DSPLs is the MAPE-K control loop, the most widely used to manage dynamic environments.

Table 13. Comparison between feature model-based approaches and the MAPE-K control loop.

Criteria	FM-Based Approaches	MAPE-K Control Loop
Allows describing possible system states	✓	
Allows to change the variability of a system at runtime	✓	✓
Allows to store the record of changes in variability		✓

Interpreting Answers for RQ3

Based on the gathered evidence for RQ3, it is possible to state that the main challenges in the management of DSPLs are based on techniques. According to [75], empirical research should be performed to obtain more realistic probability distributions of real-time constraints because most proposals are performed in a test environment, affirming what was mentioned by [13]. Also, there is a need to validate the proposals, either in an industrial environment or in different test cases, expanding the application areas of the proposed approach.

In the case of challenges related to the area of open variation, most proposals seek to improve existing approaches. For example, the proposal of Sharifloo to implement the evolution model offered by MAPE-K and automating the support for learning and evolution, defining three edges associated with the definition of safe operational limits, sandboxing, and verification of the variability associated with a self-adaptive system at runtime [39]. According to [76], it is necessary to use some mechanism to recognize new context entities and insert them into the adaptation rules.

From Figure 17, we can visualize an increase in the challenges associated with open variation and a decrease in static approaches, such as the definition of explicit variation points. Finally, we can state that the area of open dynamic variability is constantly increasing, and attempts are made to minimize the use of closed dynamic variability due to its limitations in reconfigurations. The works that do not mention challenges are related to applying an open dynamic variability.

6.1.2. Interpreting Answers to PQs

Interpreting Answers for PQ1

According to the gathered evidence to answer PQ1, we can note that from 2010 to 2018, there was a continuous growth in the number of published papers. Then, between the years 2019 to 2020, the number of articles did not exceed five per year, obtaining a notorious decrease in the year 2020, recovered in the year 2021 with seven papers. A possible interpretation could be that management of DSPLs is still booming, and the pause in the publication for 2019 and 2020 may be due to external causes, such as the pandemic generated by COVID-19. Another cause could be that the research led to more specific subtopics that we have not captured in this paper due to the limitation of the application area towards self-adaptive systems. Whether these ideas are confirmed or not, much more extensive bibliometric research is required.

Interpreting Answers for PQ2

Based on the evidence collected to answer PQ2, we can remark that more than 65% of the papers were published at conferences. The SPLC (twelve papers), SEAMS (five papers), VaMoS (four papers), and SBES (four papers) seem to be the most relevant conferences for variability management in DSPL. The rest of the conferences only registered two or one paper each. On the other hand, approximately 30% of the papers were published in a journal. The most important journals seem to be *Journal of Systems and Software* (five papers), *ACM SIGPLAN Notices* and *Science of Computer Programming* (three papers each), *IEEE Software* and *Journal of Software: Evolution and Process* (two papers each). The remains of the journals register only one paper each. Finally, the most relevant publishers were ACM, IEEE, and Science Direct. In the classification of the conferences according to the Conference ranks web site http://www.conferenceranks.com/, under the Qualis ranking (accessed May 2022). Sixteen conferences or workshops are not classified because several workshops are inactive and therefore are not indexed under the CORE or Qualis ranking. For details, see Figure 16.

6.2. Relationships between RQs

Next, we refer to how selected papers respond to each RQ. A Sankey diagram is shown in Figure 18. This diagram represents the relationships between the approaches to runtime variability management (RQ1), the applied methodology (RQ2), and the existing challenges mentioned by the authors (RQ3). The Sankey diagram allows focusing on the flows within a system.

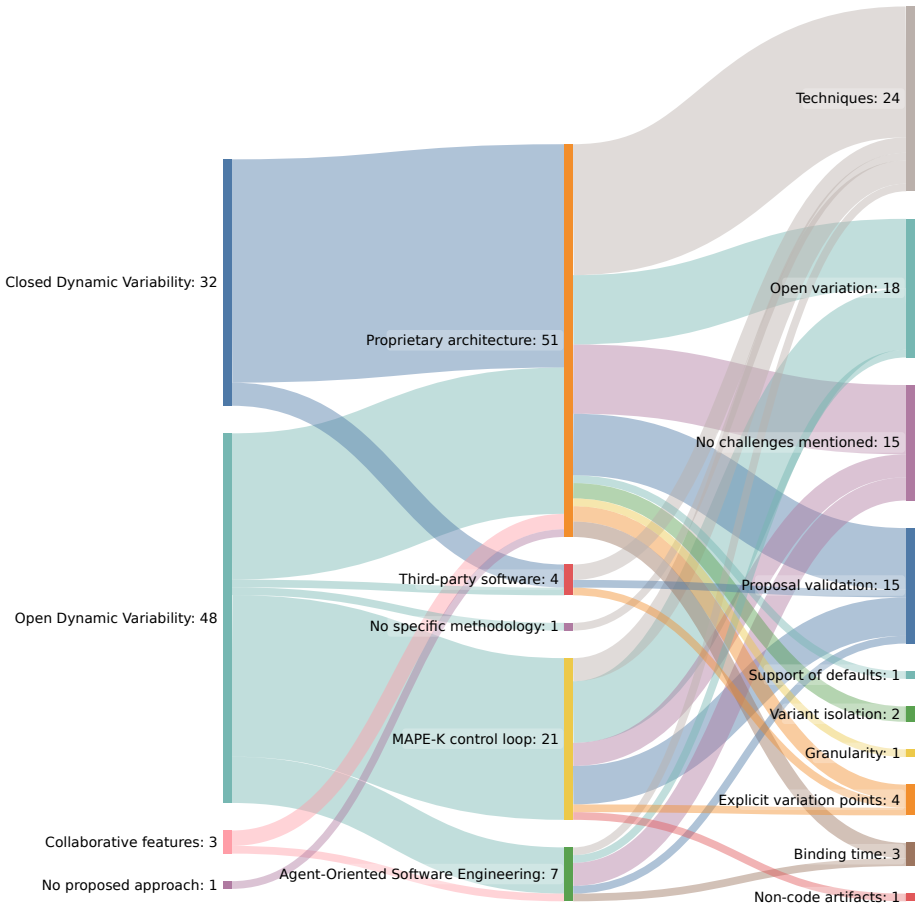

Figure 18. Relationships between approaches to runtime variability management (RQ1), applied methodology (RQ2), and challenges mentioned by the authors (RQ3).

The first vertical axis represents the approaches defined for RQ1. These approaches consider closed dynamic variability, open dynamic variability, collaborative features, and No proposed approach to manage runtime variability. The second vertical axis represents the methodology defined for RQ2. The methodologies comprise a Proprietary architecture, the MAPE-K control loop, Agent-Oriented Software Engineering, the use of Third-party software, and works with No specific methodology presented. The third vertical axis represents the challenges defined for RQ3. These challenges considered Techniques, Open Variation, Proposal Validation, Explicit Variation Points, Support of defaults, Variant Isolation, Granularity, Binding Time, Non-code Artifacts, and works that do not mention challenges.

The variability mechanisms (RQ1) and the methodology applied to manage DSPL (RQ2) are intrinsically related. Twenty-nine papers related to closed dynamic variability are managed based on a proprietary architecture, where variability modeling constraints and optimization models to improve the variant selection process predominate. To a lesser extent, three papers used third-party software. In the case of open dynamic variability, twenty-one papers considered the MAPE-K control loop. Nineteen papers are under a proprietary architecture, most of which are based on optimization models to compute the best reconfiguration that satisfies the environment requirements. Proposals managed under Agent-Oriented Software Engineering, with six papers, ending with one work associated with third-party software and non-specific methodology. The management of runtime variability through Collaborative features is addressed in two papers with their own architectures to manage a DSPL environment. One paper presents an approach based on Agent-Oriented Software Engineering. Finally, the paper that does not propose an approach to manage variability because it only presents problems and challenges exposes a theoretical model with the edges the architecture should contain.

The relationship between the methodologies for managing DSPLs (RQ2) and their challenges (RQ3) presents seventeen papers defining a Proprietary architecture. Also, these papers highlight challenges in Techniques, such as [48], that present the need to apply DSPL concepts to build adaptive multi-cloud environments, i.e., cloud systems composed of services that operate through different providers. Nine papers present challenges concerning Open variation, seeking to extend the concept of closed dynamic variability towards an open one. Proposal Validation is highlighted by eight authors, who mention that such an architecture should be validated in the industry or the application areas should be extended. To a lesser extent, there are challenges in the area of Explicit variation points, where the need to apply the context variability approach presented for the dynamic behavior of the features is exposed. It mentions that the context functions must be activated and deactivated depending on the context conditions [77]. Two papers present the need for challenges in Variant isolation and Binding times in their architectures. One architecture presents the need to apply defined granularity to variants and Support defaults. This need considers traceability and history of DSPL evolutions to support the reversal of faulty reconfigurations or to apply proactive adaptations [78]. Finally, nine architectures do not mention challenges.

The challenges for methodologies using the MAPE-K control loop include eight papers presenting challenges in Open variation. According to [7], we must consider four main problems for automatic reconfigurations using MAPE-K: the need for extensive training data, low initial performance in the case of online learning, active exploration, and the fact that real-world problems are not necessarily Markov Decision Processes (MDP). The validation of the proposal is linked to five papers. Three authors mention challenges concerning Techniques; how is the case of [51] that proposes to apply online learning algorithms to continuously refine the performance influence models to changing context specifications at runtime. To a lesser extent, there are challenges with non-coded artifacts, where [79] proposes to extend the compiler's static analysis to perform other relevant analyses, such as consistency checks and the integrity of DSL instances. Challenges are also expressed concerning explicit variation points, a proposal to improve the models by taking into account the need to add or modify variants and variation points of a feature

model in conjunction with the MAPE-K control loop [57]. Finally, three papers do not mention challenges.

The methodologies associated with Third Party Software present two challenges in the area of techniques: explicit variation points and one proposal validation. The Agent-Oriented Software Engineering, only one challenge is mentioned in the areas of Binding Time, Open Variation, Techniques, and Proposal validation. Lastly, three documents are intended to validate the proposal. Within the proposal that does not mention methodology, it presents as a challenge in the area of Techniques; the problem of the SCT language (State to Constraint Transition) occupied in the open dynamic variability approach is still related to the time requirements. This limitation comes from the inability of the constraint programming paradigm to support the various aspects of time [80].

6.3. Bibliometric Analysis

We conducted an initial bibliometric analysis of the selected papers. This analysis pretends to achieve knowledge of the most relevant terms and authors and their relationships. Figure 19 allows a first impression of the scope of the selected papers showing a simple weighted word cloud. This word cloud was generated from the titles of the selected papers, considering the fifty most relevant terms. The results align with this study's scope (e.g., adaptation, configuration, dynamic, reconfiguration, runtime, self-adaptive, and variability, among others).

Figure 19. Weighted word cloud from titles of selected papers.

In addition, the most relevant concepts were extracted from the titles of the selected papers. To do this, we used a web service called Termine, which is freely available from the academic domain of the University of Manchester, and it is based on the C-/NC-value method [81]. A total of 153 relevant terms was obtained. A top 40 list of terms is shown as an example in Table 14, ordered by relevance (score) in descending order. Through the bibliographic analysis obtained by keywords and terms, we visualize topics associated with DSPLs to model variability mainly through feature models and their subsequent adaptation through constraints. Software Product Lines is considered the most relevant term since DSPLs are a specialized area.

The results support the appropriateness of the inclusion of the selected papers in our study. Next, Figures 20 and 21 show two maps about the most relevant terms and authors. We used the VOSviewer tool to build them.

Table 14. Weighted concepts extracted from the titles of the selected papers.

Rank	Term	Score	Rank	Term	Score
1	software product line	53.12	21	component-based adaptation approach	1.58
2	dynamic software product line	49.41	21	model-based runtime adaptation	1.58
3	product line engineering	4.71	21	dynamic aspect variability	1.58
3	dynamic software product lines	4.71	21	reified contextual information	1.58
4	feature model	4	21	industrial cyber-physical system	1.58
6	adaptive software architecture	3.17	21	runtime software adaptation	1.58
7	software product line design based approach	2.58	21	adaptive fault tolerance	1.58
8	dynamic software product line approach	2.32	21	brazilian female perspective	1.58
8	maintainable dynamic software product line	2.32	21	self-adapting mobile system	1.58
8	goal-driven software product line approach	2.32	21	product derivation process	1.58
8	dynamic software product lines based	2.32	21	modeling contextual variability	1.58
12	software product line engineering	2	21	multiobjective evolutionary algorithm	1.58
12	clustering feature model based	2	21	self-adaptive software system	1.58
12	runtime variability mechanism based	2	21	multiobjective optimization algorithm	1.58
12	autonomic web service composition	2	21	efficient consistency checking	1.58
12	software product line-based approach	2	21	feature-oriented variability reconfiguration	1.58
12	software product line infrastructure	2	21	safe runtime adaptation	1.58
12	state-constraint transition modelling language	2	21	multimedia content adhering	1.58
12	cyber-physical system	2	21	dynamic adaptive system	1.58
20	feature models	1.95	40	dynamic modeling	1

6.3.1. Most Relevant Terms

Figure 20 shows the relationship between the most relevant terms for variability management in DSPL for self-adaptive systems from the keywords of the selected papers. The relevance of each term corresponds to the circle size. The color of each circle shows the evolution of terms over time. The Figure considers 6 clusters, including the 25 most relevant terms. Furthermore, we built a thesaurus to focus on specific methodological and technological concepts, unifying the terms and all their variants under a single term (e.g., terms such as runtime adaption, runtime adaptability, and runtime reconfiguration).

Through the clusters, a connection between keywords and titles can be visualized, keeping as the main axis of the DSPLs the modeling and management of runtime variability, there being a strong connection between self-adaptive systems, variability and the MAPE-K loop. Between 2010 and 2015, a connection between service-based software systems and self-adaptive systems is generated, observing topics such as constraint programming or formal languages. Between 2015 and 2017, a boom is visualized in DSPLs proposals with main focus on runtime variability and its modeling. There is a connection between DSPL proposals with the MAPE-K control loop, to manage system reconfigurations. Between 2017 and 2022, there is a connection of DSPLs with cyber-physical systems that required dynamic management of requirements, generating an evolution of DSPL application areas. Certain Software Engineering criteria are also taken into account, such as the design and development stages through reuse, touching on topics such as optimization through mathematical models to manage variability.

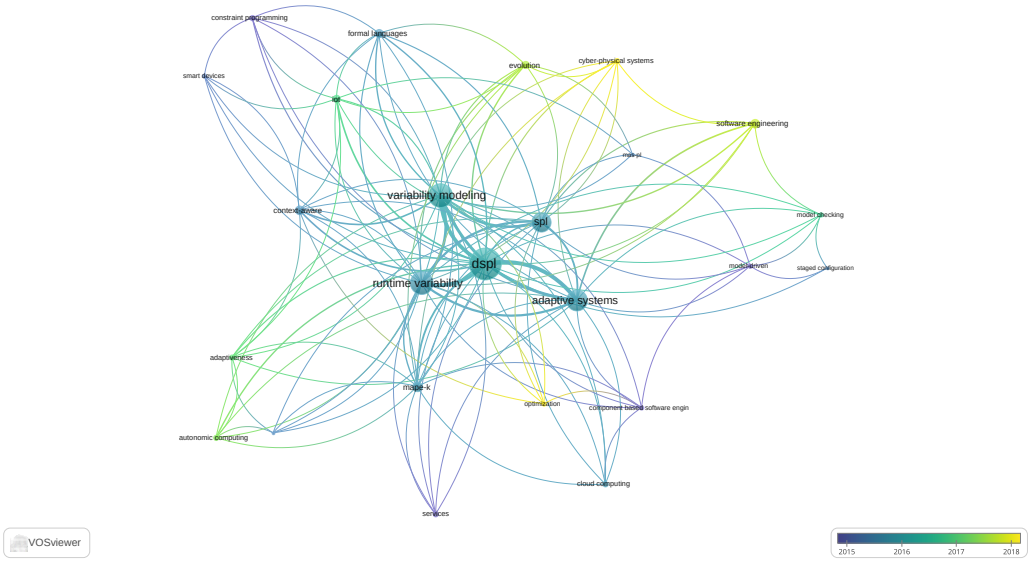

Figure 20. Relationship between the 25 most relevant terms.

6.3.2. Most Relevant Authors

Figure 21 shows the relationships between the 100 most relevant authors in the variability management in DSPL for the domain of the self-adaptive system. The size of the circles corresponds to each author's number of published papers, and their color shows the evolution of these collaborations over time. The clusters show the interactions between authors working together.

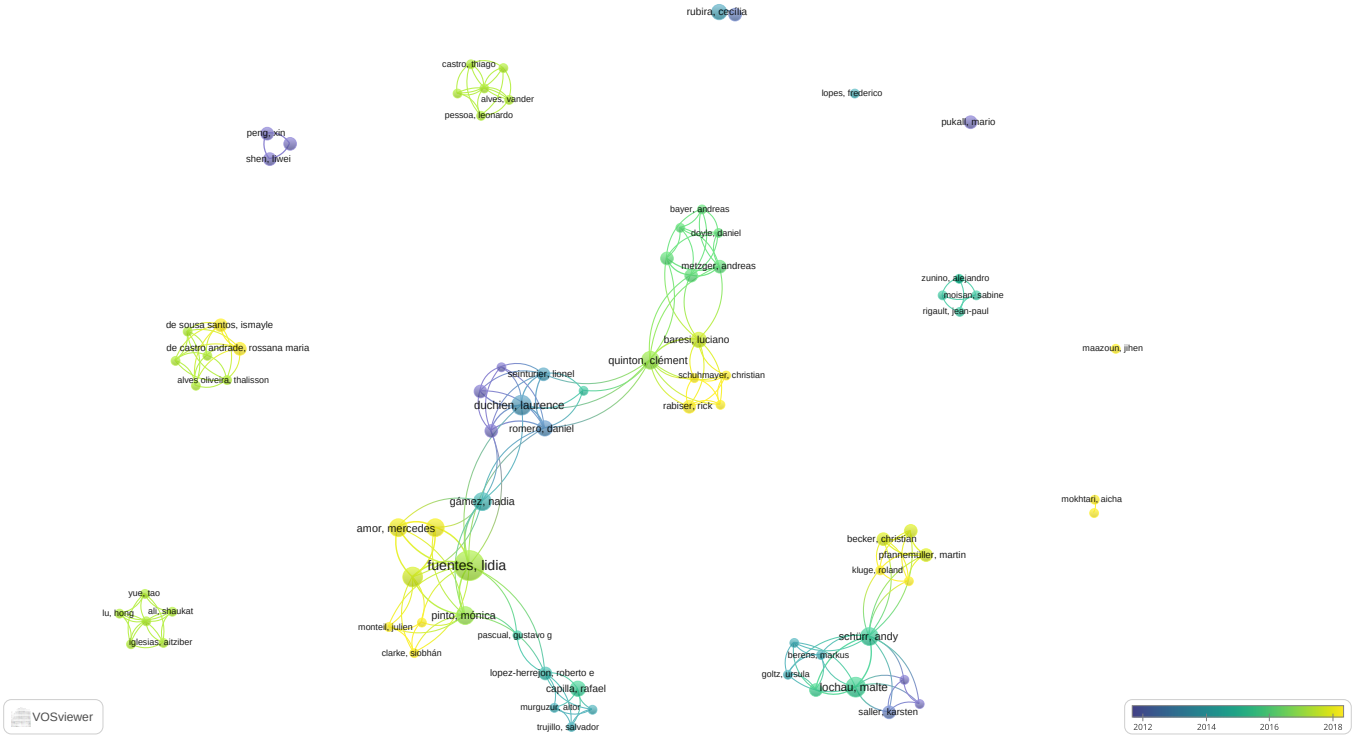

Figure 21. The most relevant authors and their relationships.

6.4. Threats to Validity

This section will discuss mitigation techniques to minimize the effects of some well-known limitations and threats to the validity of the secondary studies [82].

6.4.1. Descriptive Validity

This validity criterion ensures that observations are described objectively and accurately independent of the researcher. The mitigation techniques considered:

- The information to be collected was structured using various forms of data extraction (for RQs and PQs) through a Google Sheets data spreadsheet to support uniform data recording and ensure the objectivity of the data extraction process.
- Weekly meetings were carried out to unify critical concepts with the research and classification criteria, answer any questions and demonstrate how to carry out the process. All the researchers and assistants participated in these meetings.

6.4.2. Theoretical Validity

This validity criterion is associated with the ability to obtain the information to be captured. The associated mitigation actions considered:

- We built a search string and adapted it to the five data sources defined.
- We defined a set of exclusion and inclusion criteria to ensure objectivity in the selection process, in addition to performing cross-checks among researchers to visualize the applicability of the criteria.
- We considered that including articles written in English and discarding studies in other languages could have a minimal impact on this criterion.
- The scope of the study was expanded with a first snowballing search review, according to the guidelines provided by Wohlin [42], obtaining sixteen additional papers for the study.

6.4.3. Generalizability

This validity criterion refers to the ability to generalize the results to the entire domain. The associated mitigation actions considered:

- We assured that the scope of RQs was broad enough to identify and classify results on different DSPL approaches, regardless of specific cases, and industry type, among others.
- We used taxonomies by other authors to classify two of the three RQs.

6.4.4. Interpretive Validity

Given the data, this validity criterion is met when the study's conclusions are reasonable. The associated mitigation actions considered:

- Both researchers reviewed and validated the conclusions of the study.
- A researcher with expertise in the area of variability management in SPL assisted us in interpreting the data.

6.4.5. Repeatability

This validity criterion ensures that the research process is sufficiently detailed and that its results can be replicated comprehensively. The associated mitigation actions considered:

- We designed a detailed protocol (see Section 4), so those other researchers can repeat the whole process.
- We published the protocol through the arXiv platform [35], so those other researchers can replicate the process and corroborate the results.

7. Conclusions

This article presented a systematic mapping study on variability management in DSPLs for Self-Adaptive Systems from 2010 to 2021. We selected 84 articles that met

the inclusion and exclusion criteria. We defined three RQs to summarize the proposals according to approaches to manage runtime variability, the methodology used to manage the DSPL environment, and the current challenges generated by that application. Also, we defined two PQs to show some bibliographic characteristics of the collected papers.

We found that within the approaches to manage runtime variability, open dynamic variability predominates and, to a lesser extent, closed dynamic variability. Within the methodologies for managing DSPLs, it can be stated that the MAPE-K control loop is the backbone of most proposals associated with open dynamic variability. Concerning closed dynamic variability, some proprietary architecture is usually used, of which the generation of constraints in the modeling, through context-aware feature models, predominates. As the main areas of challenges, the need to improve techniques to manage variability stands out, either through the application of optimization models [23], validation in model consistency [79], improving the quality of service to seek more optimal reconfigurations [65], among others. Secondly, one can visualize challenges with open variation due to optimization issues in reconfigurations with open dynamic variability [51], one seeks to change from a system with closed variability to an open one. We observed that papers related to DSPLs are usually published in conferences, with a comparison of 70% versus 30% of publications in journals. The bibliographic analysis shows that the SPLC is the most relevant conference, and the Journal of Systems and Software is the most relevant journal in the area.

In future work, we plan to generate an architecture that meets the challenges mentioned in this SMS. Specifically, to generate a hybrid reconfiguration engine, which allows describing static reconfigurations in a context-aware feature model with adaptation rules and also the MAPE-K control loop, merging the closed and open dynamic variability methodologies, avoiding the problems associated with the limitation in reconfigurations and the problems mentioned by [7,83,84]. We also intend to conduct an empirical study on validating the proposals to manage variability in DSPLs, similar to [13], because we repeatedly found in this study papers mentioning the need to validate the proposal.

Author Contributions: Conceptualization, O.A. and S.S.; methodology, O.A.; validation, S.S. and O.A.; formal analysis, S.S.; investigation, O.A.; resources, O.A.; data curation, S.S.; writing—original draft preparation, O.A.; writing—review and editing, S.S.; visualization, O.A.; supervision, S.S.; project administration, S.S.; funding acquisition, S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Universidad de La Frontera, Vicerrectoría de Investigación y Postgrado together with Vicerrectoría de Pregrado. Samuel Sepúlveda thanks to research project DIUFRO IF22-0006.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Special thanks to Camila Muñoz for her helpful technical support in this work.

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

Appendix A

Table A1 presents a summary of the related work, considering the goal, RQs, time span, the number of papers included, and the main results.

Table A1. Summary of related work.

Ref.	Goal	RQs	Time Span and #Papers	Results
[9]	Provide a review on how to capture variability in DSPLs and how these models behave at runtime.	**RQ1**: How is requirements variability modelled in approaches for Dynamic software product lines? **RQ2**: How does the variability configuration process occur in dynamic software product lines?	2006–2015 #54 papers	The author states that most studies use feature models to capture the variability of DSPLs. For changes in system variability, we obtained that 12 studies occupy NFRs as a guide. However, there are 12 studies that do not specify how they would address this issue.
[31]	This research aims to identify how SPL engineering has been applied along with the IoT paradigm.	**RQ1**: How are software product lines (SPL) being applied in the context of the Internet of Things (IoT) systems? **RQ2**: How is the variability management (VM) of SPL carried out in IoT systems? **RQ3**: Which approaches, frameworks or platforms use SPL in IoT systems?	2006–2018 #56 papers	In the SPL context, through domain engineering, a problem space is described to represent customized SPL products for IoT through feature models that extend to a multi-perspective process with variation points, variants, relationships and constraints, applied in application engineering.
[32]	The study seeks to understand how dynamic derivation is performed in DSPL, by defining the inputs needed to perform dynamic derivation, describing what makes up these inputs, and understanding the process for performing dynamic derivation.	**RQ1**: What are the needed inputs to perform the dynamic derivation in DSPL? **RQ2**: How do the models, approaches and methods of software engineering address the dynamic derivation problem in DSPL? **RQ3**: Are the inputs to the dynamic derivation generated in an automatic way?	2005–2012 #20 papers	The entries for dynamic derivation can be grouped into two parts. The first, a reconfiguration plan that has information about the changes to be made at runtime. The second part are the features that are the artifacts that change at runtime, where to perform the dynamic derivation a configurator is used, where the most used is the MAPE-K loop.
[13]	To provide a review the status of the evaluation of reported Variability Management (VM) approaches and synthesize the available evidence on the effects of the reported approaches.	**RQ1**: How have the variability management approaches in SPLE been evaluated? **RQ2**: What is the quality of the reported evaluations of the variability management approaches? **RQ3**: What evidence is available about the effects of variability management approaches?	1990–2007 #97 papers	The authors state that the evaluation of the presented approaches is less rigorous from a scientific point of view, such as "application of examples", "experience report" and "discussion". This finding indicates a general lack of robust evaluation of most approaches.
[33]	To provide a study that aims to characterize and identify existing research on the use and exploitation of Software Engineering and SPL engineering.	**RQ1**: In which fora is research on integrating or combining SO and SPLE principles and practices published? **RQ2**: What is the focus and objective (motivation) of the existing research results on integrating or combining SO and SPLE principle and practices? what are the identified characteristics of the possible exploitation? **RQ3**: What are the domains and contexts applied in these proposals? **RQ4**: What types of research and contribution are represented?	2000–2011 #81 papers	The main research focus of the identified studies have been service variability modeling, service reuse, service identification, service configuration and customization, dynamic software product line and adaptive systems, where most of the studies have focused on service variability modeling and adaptive systems, applying SPLE principles and approaches. In addition, most of the studies are solution proposals (41.4%) and conceptual proposals (23.4%), with the main focus on modeling and variability management.

Appendix B

Table A2 presents some details on the selected papers, including paper ID, title, authors, publication year, source, and publisher.

Table A2. List of selected papers.

ID	Title Authors Year Source Publisher
SP1	Comparing Configuration Approaches for Dynamic Software Product Lines. Guedes G, Silva C, Soares M. 2017. SBES'17: Proceedings of the 31st Brazilian Symposium on Software Engineering.
SP2	Learning and Evolution in Dynamic Software Product Lines. Sharifloo AM, Metzger A, Quinton C, Baresi L, Pohl K, 2016, SEAMS, ACM.
SP3	Optimal Reconfiguration of Dynamic Software Product Lines Based on Performance-Influence Models. Weckesser M, Kluge R, Pfannemüller M, Matthé M, Schürr A, Becker C, 2018, SPLC, ACM.
SP4	Tailoring Dynamic Software Product Lines Rosenmüller M, Siegmund N, Pukall M, Apel S, 2011, ACM-SN, ACM.
SP5	Model Verification of Dynamic Software Product Lines Santos IS, Rocha LS, Neto PA, Andrade RM, 2016, SBES, ACM.
SP6	Extending Dynamic Software Product Lines with Temporal Constraints Sousa G, Rudametkin W, Duchien L, 2017, SEAMS, IEEE.
SP7	Using Document-Oriented GUIs in Dynamic Software Product Lines Kramer D, Oussena S, Komisarczuk P, Clark T, 2013, ACM-SN, ACM.
SP8	Towards Autonomic Software Product Lines. Abbas N, 2011, SPLC, ACM.
SP9	Trace Checking for Dynamic Software Product Lines Olaechea R, Atlee J, Legay A, Fahrenberg U, 2018, SEAMS, ACM.
SP10	Executable Modelling of Dynamic Software Product Lines in the ABS Language Muschevici R, Clarke D, Proença J, 2013, FOSD, ACM.
SP11	Context Variability Modeling for Runtime Configuration of Service-Based Dynamic Software Product Lines. Murguzur A, Capilla R, Trujillo S, Ortiz Ó, Lopez-Herrejon RE, 2014, SPLC, ACM.
SP12	Staged Configuration of Dynamic Software Product Lines with Complex Binding Time Constraints. Bürdek J, Lity S, Lochau M, Berens M, Goltz U, Schürr A, 2014, VaMoS, ACM.
SP13	Dynamically Evolving the Structural Variability of Dynamic Software Product Lines. Baresi L, Quinton C, 2015, SEAMS, IEEE.
SP14	N-Dimensional Tensor Factorization for Self-Configuration of Software Product Lines at Runtime. Pereira JA, Schulze S, Figueiredo E, Saake G, 2018, SPLC, ACM.
SP15	A Formal Foundation for Dynamic Delta-Oriented Software Product Lines. Damiani F, Padovani L, Schaefer I, 2012, GPCE, ACM.
SP16	Product Line Engineering of Monitoring Functionality in Industrial Cyber-Physical Systems: A Domain Analysis. Iglesias A, Lu H, Arellano C, Yue T, Ali S, Sagardui G, 2017, SPLC, ACM.
SP17	Using Reified Contextual Information for Safe Run-Time Adaptation of Software Product Lines. Sunkle S, Pukall M, 2010, RAM-SE, ACM.
SP18	Towards a Software Product Line-Based Approach to Adapt IaaS Cloud Configurations. Ruiz C, Duran-Limon HA, Parlavantzas N, 2016, UCC, ACM.
SP19	Dynamic Software Adaptation for Service-Oriented Product Lines. Gomaa H, Hashimoto K, 2011, SPLC, ACM.
SP20	Constraint-Based Self-Adaptation of Wireless Sensor Networks. Gamez N, Romero D, Fuentes L, Rouvoy R, Duchien L, 2012, WAS4FI-Mashups, ACM.
SP21	Reducing Feature Models to Improve Runtime Adaptivity on Resource Limited Devices. Saller K, Oster S, Schürr A, Schroeter J, Lochau M, 2012, SPLC, ACM.
SP22	Evaluation of the State-Constraint Transition Modelling Language: A Goal Question Metric Approach. Achtaich A, Roudies O, Souissi N, Salinesi C, Mazo R, 2019, SPLC, ACM.
SP23	An SPL Approach for Adaptive Fault Tolerance in SOA. Nascimento AS, Rubira CM, Lee J, 2011, SPLC, ACM.
SP24	Dynamic Variability Management Supporting Operational Modes of a Power Plant Product Line. Capilla R, Bosch J, 2016, VaMoS, ACM.

Table A2. *Cont.*

ID	Title Authors Year Source Publisher
SP25	Using Dynamic Adaptive Systems in Safety-Critical Domains. McGee ET, McGregor JD, 2016, SEAMS, ACM.
SP26	Using Constraint-Based Optimization and Variability to Support Continuous Self-Adaptation. Parra C, Romero D, Mosser S, Rouvoy R, Duchien L, Seinturier L, 2012, SAC, ACM.
SP27	Context-Aware DSPLs: Model-Based Runtime Adaptation for Resource-Constrained Systems. Saller K, Lochau M, Reimund I, 2013, SPLC, ACM.
SP28	Research Contributions on Adaptive Software Architectures: A Brazilian Female Perspective at UNICAMP. Venero SK, Eleutério JD, Rubira CM, 2016, ECSAW, ACM.
SP29	Coordinated Run-Time Adaptation of Variability-Intensive Systems: An Application in Cloud Computing. Metzger A, Bayer A, Doyle D, Sharifloo AM, Pohl K, Wessling F, 2016, VACE, ACM.
SP30	Runtime Monitoring of Behavioral Properties in Dynamically Adaptive Systems. dos Santos EB, de Castro Andrade RM, de Sousa Santos I, 2019, SBES, ACM.
SP31	Modeling Dynamic Adaptations Using Augmented Feature Models. Jean-Baptiste L, Maria-Teresa S, Jean-Marie G, Antoine B, 2013, SAC, ACM.
SP32	Safe Adaptation in Context-Aware Feature Models. Marinho FG, Maia PH, Andrade RM, Vidal VM, Costa PA, Werner C, 2012, FOSD, ACM.
SP33	Towards an Architecture Model for Dynamic Software Product Lines Engineering. Santos E, Machado I, 2018, IRI, IEEE.
SP34	An Approach to Clustering Feature Model Based on Adaptive Behavior for Dynamic Software Product Line. Boonon P, Muenchaisri P, 2014, ICISA, IEEE.
SP35	Reconfiguration of Service Failures in DAMASCo Using Dynamic Software Product Lines. Cubo J, Gamez N, Pimentel E, Fuentes L, 2015, SCC, IEEE.
SP36	Achieving Knowledge Evolution in Dynamic Software Product Lines. Arcega L, Font J, Haugen , Cetina C, 2016, SANER, IEEE.
SP37	A Runtime Variability Mechanism Based on Supertypes. Capilla R, Valdezate A, Díaz F, 2016, FAS*W, IEEE.
SP38	Creating adaptive software architecture dynamically for recurring new requirements. Ali N, Hong J, 2017, ICOSST, IEEE.
SP39	Combining variability, RCA and feature model for context-awareness. Amja A, Obaid A, Mili H, 2016, INTECH, IEEE.
SP40	Towards a DSPL for Context Aware BPM. Khiari B, ; Jilani L, 2017, AICCSA, IEEE.
SP41	Dynamic Constraint Satisfaction Algorithm for Online Feature Model Reconfiguration. Entekhabi S, Karataş A, Oğuztüzün H, 2018, CEIT, IEEE.
SP42	Dynamic adaptation and reconfiguration of security in mobile devices. Amoud M, Roudies O, 2017, Cyber Incident, IEEE.
SP43	Dynamic SPL and Derivative Development with Uncertainty Management for DevOps. Nakanishi T, Furusho H, Hisazumi K, Fukuda A, 2016, IIAI-AAI, IEEE.
SP44	Towards Feature-Oriented Variability Reconfiguration in Dynamic Software Product Lines. ShenXin L, Zhao P, 2011, ICSR, Springer.
SP45	Designing a Framework for Smart IoT Adaptations. Achtaich A, Souissi N, Mazo R, Salinesi C, Roudies O, 2018, AFRICATEK, Springer.
SP46	SmartyCo: Managing Cyber-Physical Systems for Smart Environments. Romero D, Quinton C, Duchien L, Seinturier L, Valdez C, 2015, ECSA, Springer.
SP47	Using Models at Runtime to Adapt Self-managed Agents for the IoT. Ayala I, Horcas JM, Amor M, Fuentes L, 2016, MATES, Springer.
SP48	Autonomic Adaptation of Multimedia Content Adhering to Application Mobility. Velázquez-García F, Halvorsen P, Stensland H, Eliassen F, 2018, DAIS, Springer.
SP49	Specification and automated validation of staged reconfiguration processes for dynamic software product lines. Lochau M, Bürdek J, Hölzle S, Schürr A, 2017, SoSyM, Springer.
SP50	An approach based on feature models and quality criteria for adapting component-based systems. Sanchez L, Diaz-Pace J, Zunino A, Moisan S, Rigault J, 2015, JSERD, Springer.
SP51	Context-dependent reconfiguration of autonomous vehicles in mixed traffic. Horcas JM, Monteil J, Bouroche M, Pinto M, Fuentes L, Clarke S, 2018, SEaP, Wiley.

Table A2. *Cont.*

ID	Title Authors Year Source Publisher
SP52	ASPLe: A methodology to develop self-adaptive software systems with systematic reuse. Abbas N, Andersson J, Weyns D, 2020, JSS, Science Direct.
SP53	Automating the product derivation process of multi-agent systems product lines. Cirilo E, Nunes I, Kulesza U, Lucena C, 2012, JSS, Science Direct.
SP54	Unifying design and runtime software adaptation using aspect models. Parra C, Blanc X, Cleve A, Duchien L, 2011, SoCP, Science Direct.
SP55	Prototyping Dynamic Software Product Lines to evaluate run-time reconfigurations. Cetina C, Giner P, Fons J, Pelechano V, 2013, SoCP, Science Direct.
SP56	A goal-driven software product line approach for evolving multi-agent systems in the Internet of Things. Ayala I, Amor M, Horcas JM, Fuentes L, 2019, KBS, Science Direct.
SP57	Achieving autonomic Web service compositions with models at runtime. Alférez GH, Pelechano V, 2017, C&EE, Science Direct.
SP58	Building reliable and maintainable Dynamic Software Product Lines: An investigation in the Body Sensor Network domain. Pessoa L, Fernandes P, Castro T, Alves V, Rodrigues GN, Carvalho H, 2017, IST, Science Direct.
SP59	Context-aware reconfiguration in evolving software product lines. Mauro J, Nieke M, Seidl C, Chieh Yu I, 2018, SoCP, Science Direct.
SP60	Applying multiobjective evolutionary algorithms to dynamic software product lines for reconfiguring mobile applications. Pascual GG, Lopez-Herrejon RE, Pinto M, Fuentes L, Egyed A, 2015, JSS, Science Direct.
SP61	Context-aware energy-efficient applications for cyber-physical systems. Horcas JM, Pinto M, Fuentes L, 2019, AHN, Science Direct.
SP62	Self-adaptation of service compositions through product line reconfiguration. Bashari M, Bagheri E, Du W, 2018, JSS, Science Direct.
SP63	DyMMer 2.0: A Tool for Dynamic Modeling and Evaluation of Feature Model. Bezerra C, Lima R, Silva P, 2021, SBES, ACM.
SP64	Static Analysis Techniques for Efficient Consistency Checking of Real-Time-Aware DSPL Specifications. Göttmann H, Bacher I, Gottwald N, Lochau M, 2021, VaMoS, ACM.
SP65	ProDSPL: Proactive self-adaptation based on Dynamic Software Product Lines. Ayala I, Papadopoulos AV, Amor M, Fuentes L, 2021, JSS, Science Direct.
SP66	Towards Mastering Variability in Software-Intensive Cyber-Physical Production Systems. Rabiser R, Zoitl A, 2021, PCS, Science Direct.
SP67	A Software Product Line Design Based Approach for Real-time Scheduling of Reconfigurable Embedded Systems. Gharsellaoui H, Maazoun J, Bouassida N, Ahmed SB, Ben-Abdallah H, 2021, CHB, Science Direct.
SP68	Evolution in dynamic software product lines. Quinton C, Vierhauser M, Rabiser R, Baresi L, Grünbacher P, Schuhmayer C, 2021, SEaP, Wiley.
SP69	Transfer learning for multiobjective optimization algorithms supporting dynamic software product lines. Ballesteros J, Fuentes L, 2021, SPLC, ACM.
SP70	Dynamically Adaptable Software Is All about Modeling Contextual Variability and Avoiding Failures. de Sousa Santos I, de Jesus Souza M, Luciano Carvalho M, Alves Oliveira T, de Almeida E and de Castro Andrade R, 2017, IEEE-S, IEEE.
SP71	A study on dynamic aspects variability in the SOLAR educational software ecosystem. Coutinho E, Bezerra C, 2020, JBCS, Springer.
SP72	Variable Recovery and Adaptation Connectors for Dynamic Software Product Lines. Albassam E, Gomaa H and Menascé D, 2017, SPLC, ACM.
SP73	Creating Self-Adapting Mobile Systems with Dynamic Software Product Lines. Gamez N, Fuentes L, Troya J, 2015, IEEE-S, IEEE.
SP74	A Matter of the Mix: Integration of Compile and Runtime Variability. Eichelberger H, 2016, FAS*W, IEEE.
SP75	ArCMAPE: A Software Product Line Infrastructure to Support Fault-Tolerant Composite Services. Nascimento A, Rubira C, Castor F, 2014, HASE, IEEE.
SP76	Towards a MAS Product Line Engineering Approach. Boufedji D, Guessoum Z, Brandão A, Ziadi T, Mokhtari A, 2018, EMAS, Springer.

Table A2. *Cont.*

ID	Title Authors Year Source Publisher
SP77	Dynamic Software Product Line Engineering: A Reference Framework. Bashari M, Bagheri E, Du W, 2017, IJSEKE, World Scientific.
SP78	Using dynamic software product lines to implement adaptive SGX-enabled systems. Krieter S, Thiem T, Leich T, 2019, VaMoS, ACM.
SP79	A Dynamic Software Product Line Approach for Adaptation Planning in Autonomic Computing Systems. Pfannemüller M, Krupitzer C, Weckesser M, Becker C, 2017, ICAC, IEEE.
SP80	A component-based adaptation approach for multi-cloud applications. Almeida A, Cavalcante E, Batista T, Cacho N, Lopes F, 2014, INFOCOM WKSHPS, IEEE.
SP81	A framework for context-aware self-adaptive mobile applications SPL. Mizouni R, Abu Matar M, Al Mahmoud Z, Alzahmi S, Salah A, 2014, ESA, Science Direct.
SP82	Software Product Line Engineering for Developing Self-Adaptive Systems: Towards the Domain Requirements. Shen L, Peng X, Zhao W, 2012, COMPSAC, IEEE.
SP83	Using constraint programming to manage configurations in self-adaptive systems. Sawyer P, Mazo R, Diaz D, Salinesi C, Hughes D, 2012, Computer, IEEE.
SP84	Dynamic Reconfiguration of Security Policies in Wireless Sensor Networks. Pinto M, Gámez N, Fuentes L, Amor M, Horcas JM, Ayala I, 2015, Sensors, MDPI.

Appendix C

Tables A3 and A4 presents the acronyms used in the study to indicate conferences and journals.

Table A3. List of conferences and workshops.

Acronym	Conference/Workshop Title
AFRICATEK	International Conference on Emerging Technologies for Developing Countries
AICCSA	ACS/IEEE International Conference on Computer Systems and Applications
CEIT	International Conference on Control, Engineering & Information Technology
COMPSAC	Annual International Computer Software and Applications Conference
Cyber Incident	International Conference on Cyber Incident Response, Coordination, Containment & Control
ECSA	European Conference on Software Architecture
ECSAW	European Conference on Software Architecture Workshops
EMAS	International Workshop on Engineering Multi-Agent Systems
DAIS	IFIP International Conference on Distributed Applications and Interoperable Systems
FAS*W	International Workshops on Foundations and Applications of Self Systems
FOSD	International Workshop on Feature-Oriented Software Development
HASE	International Symposim on High Assurance Systems Engineering
ICAC	International Conference on Autonomic Computing
ICISA	International Conference on Information Science and Applications
ICOSST	International Conference on Open Source Systems and Technologies
ICSR	International Conference on Software Reuse
IIAI-AAI	IIAI International Conference on Advanced Applied Informatics
INFOCOM WKSHPS	Conference on Computer Communications Workshops
INTECH	International Conference on Innovative Computing Technology
IRI	International Conference on Information Reuse and Integration
MATES	German Conference on Multiagent System Technologies
RAM-SE	Workshop on Reflection, AOP and Meta-Data for Software Evolution
SAC	Annual ACM Symposium on Applied Computing
SANER	International Conference on Software Analysis, Evolution and Reengineering
SBES	Brazilian Symposium on Software Engineering
SCC	International Conference on Services Computing
SEAMS	Symposium on Software Engineering for Adaptive and Self-Managing Systems
SPLC	International Systems and Software Product Line Conference
VACE	International Workshop on Variability and Complexity in Software Design
VaMoS	International Workshop on Variability Modelling of Software-Intensive Systems
UCC	International Conference on Utility and Cloud Computing
WAS4FI-Mashups	International Workshop on Adaptive Services for the Future Internet and International Workshop on Web APIs and Service

Table A4. List of Journals

Acronym	Journal Title
ACM-SN	ACM SIGPLAN Notices
AHN	Ad Hoc Networks
CHB	Computers in Human Behavior
COM	IEEE Computer Society
C&EE	Computers & Electrical Engineering
ESA	Expert Systems with Applications
IEEE-S	IEEE Software
IJSEKE	International Journal of Software Engineering and Knowledge Engineering
IST	Information and Software Technology
JBCS	Journal of the Brazilian Computer Society
JSERD	Journal of Software Engineering Research and Development
JSS	Journal of Systems and Software
KBS	Knowledge-Based Systems
PCS	Procedia Computer Science
SEaP	Journal of Software: Evolution and Process
SEN	Sensors
SoCP	Science of Computer Programming
SoSyM	Software & Systems Modeling

References

1. Weyns, D. Software Engineering of Self-adaptive Systems. In *Handbook of Software Engineering*; Springer International Publishing: Cham, Switzerland, 2019; pp. 399–443. [CrossRef]
2. Hinchey, M.; Park, S.; Schmid, K. Building Dynamic Software Product Lines. *Computer* **2012**, *45*, 22–26. [CrossRef]
3. Quinton, C.; Vierhauser, M.; Rabiser, R.; Baresi, L.; Grünbacher, P.; Schuhmayer, C. Evolution in dynamic software product lines. *J. Softw. Evol. Process.* **2021**, *33*, e2293. [CrossRef]
4. Hallsteinsen, S.; Hinchey, M.; Park, S.; Schmid, K. Dynamic Software Product Lines. *Computer* **2008**, *41*, 93–95. [CrossRef]
5. Schmid, K.; Eichelberger, H. From Static to Dynamic Software Product Lines. In Proceedings of the SPLC 2008, Limerick, Ireland, 8–12 September 2008; pp. 33–38.
6. Kephart, J.; Chess, D. The vision of autonomic computing. *Computer* **2003**, *36*, 41–50. [CrossRef]
7. Abbas, N. Towards autonomic software product lines. In Proceedings of the 15th International Software Product Line Conference, Munich, Germany, 21–26 August 2011; Volume 2, pp. 1–8.
8. Cravero, A.; Pardo, S.; Sepúlveda, S.; Muñoz, L. Challenges to Use Machine Learning in Agricultural Big Data: A Systematic Literature Review. *Agronomy* **2022**, *12*, 748. [CrossRef]
9. Guedes, G.; Silva, C.; Soares, M.; Castro, J. Variability Management in Dynamic Software Product Lines: A Systematic Mapping. In Proceedings of the 2015 IX Brazilian Symposium on Components, Architectures and Reuse Software, Belo Horizonte, Minas Gerais, Brazil, 21–22 September 2015; pp. 90–99. [CrossRef]
10. Clements, P.; Northrop, L. *Software Product Lines, Course Notes of Product Line Systems Program*; Software Engineering Institute, Carnegie Mellon University: Pittsburgh, PA, USA, 2003.
11. Apel, S.; Batory, D.; Kästner, C.; Saake, G. Software Product Lines. In *Feature-Oriented Software Product Lines: Concepts and Implementation*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 3–15. [CrossRef]
12. Deelstra, S.; Sinnema, M.; Bosch, J. Experiences in Software Product Families: Problems and Issues During Product Derivation. In *Proceedings of the Software Product Lines*; Nord, R.L., Ed.; Springer: Berlin/Heidelberg, Germany, 2004; pp. 165–182.
13. Chen, L.; Ali Babar, M. A systematic review of evaluation of variability management approaches in software product lines. *Inf. Softw. Technol.* **2011**, *53*, 344–362. [CrossRef]
14. Asikainen, T.; Mannisto, T.; Soininen, T. A unified conceptual foundation for feature modelling. In Proceedings of the 10th International Software Product Line Conference (SPLC'06), Baltimore, MD, USA, 21–24 August 2006; pp. 31–40. [CrossRef]
15. Pohl, K.; Böckle, G.; Van Der Linden, F. *Software Product Line Engineering*; Springer: Berlin/Heidelberg, Germany, 2005; Volume 10.
16. Kang, K.C.; Cohen, S.G.; Hess, J.A.; Novak, W.E.; Peterson, A.S. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*; Technical Report; Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst: Pittsburgh, PA, USA, 1990.
17. Heradio, R.; Perez-Morago, H.; Fernandez-Amoros, D.; Cabrerizo, F.J.; Herrera-Viedma, E. A bibliometric analysis of 20 years of research on software product lines. *Inf. Softw. Technol.* **2016**, *72*, 1–15. [CrossRef]
18. Gacitúa, R.; Sepúlveda, S.; Mazo, R. FM-CF: A framework for classifying feature model building approaches. *J. Syst. Softw.* **2019**, *154*, 1–21. [CrossRef]
19. Sepúlveda, S.; Cravero, A. Reasoning Algorithms on Feature Modeling—A Systematic Mapping Study. *Appl. Sci.* **2022**, *12*, 5563. [CrossRef]
20. Kim, M.; Park, S. Goal and scenario driven product line development. In Proceedings of the 11th Asia-Pacific Software Engineering Conference, Washington, DC, USA, 30 November–3 December 2004; pp. 584–585. [CrossRef]

21. Fitzgerald, B.; Stol, K.J. Continuous Software Engineering and beyond: Trends and Challenges. In Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering (RCoSE 2014), Hyderabad, India, 3 June 2014; Association for Computing Machinery: New York, NY, USA, 2014; pp. 1–9. [CrossRef]
22. Rosenmüller, M.; Siegmund, N.; Pukall, M.; Apel, S. Tailoring dynamic software product lines. In Proceedings of the 10th ACM International Conference on Generative Programming and Component Engineering, Portland, OR, USA, 22–23 October 2011; pp. 3–12.
23. Santos, I.S.; Rocha, L.S.; Neto, P.A.S.; Andrade, R.M. Model verification of dynamic software product lines. In Proceedings of the 30th Brazilian Symposium on Software Engineering, Maringá, Brazil, 19–23 September 2016; pp. 113–122.
24. Guedes, G.; Silva, C.; Soares, M. Comparing configuration approaches for dynamic software product lines. In Proceedings of the 31st Brazilian Symposium on Software Engineering, Fortaleza, CE, Brazil, 20–22 September 2017; pp. 134–143.
25. Salehie, M.; Tahvildari, L. Self-Adaptive Software: Landscape and Research Challenges. *ACM Trans. Auton. Adapt. Syst.* **2009**, *4*, 1–42. [CrossRef]
26. de Lemos, R.; Giese, H.; Müller, H.A.; Shaw, M.; Andersson, J.; Litoiu, M.; Schmerl, B.; Tamura, G.; Villegas, N.M.; Vogel, T.; et al. Software Engineering for Self-Adaptive Systems: A Second Research Roadmap. In *Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, 24–29 October 2010 Revised Selected and Invited Papers*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 1–32. [CrossRef]
27. Bosch, J.; Capilla, R. Dynamic Variability in Software-Intensive Embedded System Families. *Computer* **2012**, *45*, 28–35. [CrossRef]
28. Galster, M.; Weyns, D.; Tofan, D.; Michalik, B.; Avgeriou, P. Variability in software systems—A systematic literature review. *IEEE Trans. Softw. Eng.* **2013**, *40*, 282–306. [CrossRef]
29. Raatikainen, M.; Tiihonen, J.; Männistö, T. Software product lines and variability modeling: A tertiary study. *J. Syst. Softw.* **2019**, *149*, 485–510. [CrossRef]
30. Jaffari, A.; Lee, J.; Kim, E. Variability Modeling in Software Product Line: A Systematic Literature Review. In *Software Engineering in IoT, Big Data, Cloud and Mobile Computing*; Springer: Cham, Switzerland, 2021; pp. 1–15.
31. Geraldi, R.T.; Reinehr, S.; Malucelli, A. Software product line applied to the internet of things: A systematic literature review. *Inf. Softw. Technol.* **2020**, *124*, 106293. [CrossRef]
32. da Silva, J.R.F.; da Silva, F.A.P.; do Nascimento, L.M.; Martins, D.A.O.; Garcia, V.C. The dynamic aspects of product derivation in DSPL: A systematic literature review. In Proceedings of the 2013 IEEE 14th International Conference on Information Reuse and Integration (IRI), San Francisco, CA, USA, 14–18 August 2013; pp. 466–473. [CrossRef]
33. Mohabbati, B.; Asadi, M.; Gašević, D.; Hatala, M.; Müller, H.A. Combining service-orientation and software product line engineering: A systematic mapping study. *Inf. Softw. Technol.* **2013**, *55*, 1845–1859. [CrossRef]
34. Petersen, K.; Vakkalanka, S.; Kuzniarz, L. Guidelines for conducting systematic mapping studies in software engineering: An update. *Inf. Softw. Technol.* **2015**, *64*, 1–18. [CrossRef]
35. Aguayo, O.; Sepúlveda, S. Systematic Mapping Protocol: Variability Management in Dynamic Software Product Lines for Self-Adaptive Systems. *arxiv* **2022**. [CrossRef]
36. Kitchenham, B.; Charters, S.; Budgen, S.; Brereton, P.; Turner, M.; Linkman, S.; Jørgensen, M.; Mendes, E.; Visaggio, G. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*; EBSE Technical Report, EBSE-2007-01; Software Engineering Group, School of Computer Science and Mathematics, Keele University: Keele, UK, 2007.
37. Brereton, P.; Kitchenham, B.A.; Budgen, D.; Turner, M.; Khalil, M. Lessons from applying the systematic literature review process within the software engineering domain. *J. Syst. Softw.* **2007**, *80*, 571–583. [CrossRef]
38. Petticrew, M.; Roberts, H. How to Find the Studies: The Literature Search. In *Systematic Reviews in the Social Sciences*; John Wiley & Sons, Ltd.: New York, NY, USA, 2006; Chapter 4, pp. 79–124. [CrossRef]
39. Sharifloo, A.M.; Metzger, A.; Quinton, C.; Baresi, L.; Pohl, K. Learning and evolution in dynamic software product lines. In Proceedings of the 2016 IEEE/ACM 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), Austin, TX, USA, 16–17 May 2016; pp. 158–164.
40. Arcega, L.; Font, J.; Haugen, Ø.; Cetina, C. Achieving knowledge evolution in dynamic software product lines. In Proceedings of the 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Suita, Osaka, Japan, 14–18 March 2016; Volume 1, pp. 505–516.
41. Gusenbauer, M.; Haddaway, N.R. Which academic search systems are suitable for systematic reviews or meta-analyses? Evaluating retrieval qualities of Google Scholar, PubMed, and 26 other resources. *Res. Synth. Methods* **2020**, *11*, 181–217. [CrossRef]
42. Wohlin, C. Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering. In Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE '14), London, UK, 13–14 May 2014; Association for Computing Machinery: New York, NY, USA, 2014. [CrossRef]
43. Gwet, K. Inter-rater reliability: Dependency on trait prevalence and marginal homogeneity. *Stat. Methods Inter-Rater Reliab. Assess. Ser.* **2002**, *2*, 9.
44. Fleiss, J.L. *Statistical Methods for Rates and Proportions*, 2nd ed.; Probability & Mathematical Statistics S.; John Wiley & Sons: Nashville, TN, USA, 1981.
45. Ancán, O.; Reyes, M. *Cabuplot: Categorical Bubble Plot for Systematic Mapping Studies*; Departamento de Ciencias de la Computación e Informática, Universidad de La Frontera, Temuco, Chile, 2020.

46. Mens, K.; Capilla, R.; Cardozo, N.; Dumas, B. A Taxonomy of Context-Aware Software Variability Approaches. In *MODULARITY Companion 2016, Proceedings of the 15th International Conference on Modularity, Málaga, Spain, 14–17 March 2016*; Association for Computing Machinery: New York, NY, USA, 2016; pp. 119–124. [CrossRef]
47. Zhang, B.; Duszynski, S.; Becker, M. Variability Mechanisms and Lessons Learned in Practice. In Proceedings of the 1st International Workshop on Variability and Complexity in Software Design (VACE '16), Austin, TX, USA, 15 May 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 14–20. [CrossRef]
48. Sousa, G.; Rudametkin, W.; Duchien, L. Extending dynamic software product lines with temporal constraints. In Proceedings of the 2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), Buenos Aires, Argentina, 22–23 May 2017; pp. 129–139.
49. Ruiz, C.; Duran-Limon, H.A.; Parlavantzas, N. Towards a software product line-based approach to adapt IaaS cloud configurations. In Proceedings of the 9th International Conference on Utility and Cloud Computing, Shanghai, China, 6–9 December 2016; pp. 398–403.
50. Kramer, D.; Oussena, S.; Komisarczuk, P.; Clark, T. Using document-oriented GUIs in dynamic software product lines. *ACM SIGPLAN Notices* **2013**, *49*, 85–94. [CrossRef]
51. Weckesser, M.; Kluge, R.; Pfannemüller, M.; Matthé, M.; Schürr, A.; Becker, C. Optimal reconfiguration of dynamic software product lines based on performance-influence models. In Proceedings of the 22nd International Systems and Software Product Line Conference, Gothenburg, Sweden, 10–14 September 2018; Volume 1, pp. 98–109.
52. Nascimento, A.S.; Rubira, C.M.F.; Lee, J. An spl approach for adaptive fault tolerance in soa. In Proceedings of the 15th International Software Product Line Conference, Munich, Germany, 21–26 August 2011; Volume 2, pp. 1–8.
53. Venero, S.K.; Eleutério, J.D.; Rubira, C.M. Research contributions on adaptive software architectures: A Brazilian female perspective at UNICAMP. In Proceedings of the 10th European Conference on Software Architecture Workshops, Copenhagen, Denmark, 28 November–2 December 2016; pp. 1–6.
54. dos Santos, E.B.; de Castro Andrade, R.M.; de Sousa Santos, I. Runtime monitoring of behavioral properties in dynamically adaptive systems. In Proceedings of the XXXIII Brazilian Symposium on Software Engineering, Salvador, Brazil, 23–27 September 2019; pp. 377–386.
55. Santos, E.; Machado, I. Towards an architecture model for dynamic software product lines engineering. In Proceedings of the 2018 IEEE International Conference on Information Reuse and Integration (IRI), Salt Lake City, UT, USA, 6–9 July 2018; pp. 31–38.
56. Boonon, P.; Muenchaisri, P. An approach to clustering feature model based on adaptive behavior for dynamic software product line. In Proceedings of the 2014 IEEE International Conference on Information Science & Applications (ICISA), Seoul, Korea, 6–9 May 2014; pp. 1–4.
57. Amja, A.M.; Obaid, A.; Mili, H. Combining variability, RCA and feature model for context-awareness. In Proceedings of the 2016 Sixth IEEE International Conference on Innovative Computing Technology (INTECH), Dublin, Ireland, 24–26 August 2016; pp. 15–23.
58. Khiari, B.; Jilani, L.L. Towards a DSPL for Context Aware BPM. In Proceedings of the 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), Hammamet, Tunisia, 30 October–3 November 2017; pp. 12–18.
59. Amoud, M.; Roudies, O. Dynamic adaptation and reconfiguration of security in mobile devices. In Proceedings of the 2017 IEEE International Conference On Cyber Incident Response, Coordination, Containment & Control (Cyber Incident), London, UK, 19–20 June 2017; pp. 1–6.
60. Velázquez-García, F.J.; Halvorsen, P.; Stensland, H.K.; Eliassen, F. Autonomic adaptation of multimedia content adhering to application mobility. In Proceedings of the IFIP International Conference on Distributed Applications and Interoperable Systems, Madrid, Spain, 18–21 June 2018; Springer: Berlin, Germany, 2018; pp. 153–168.
61. Abbas, N.; Andersson, J.; Weyns, D. ASPLe: A methodology to develop self-adaptive software systems with systematic reuse. *J. Syst. Softw.* **2020**, *167*, 110626. [CrossRef]
62. Cirilo, E.; Nunes, I.; Kulesza, U.; Lucena, C. Automating the product derivation process of multi-agent systems product lines. *J. Syst. Softw.* **2012**, *85*, 258–276. [CrossRef]
63. Ayala, I.; Amor, M.; Horcas, J.M.; Fuentes, L. A goal-driven software product line approach for evolving multi-agent systems in the Internet of Things. *Knowl.-Based Syst.* **2019**, *184*, 104883. [CrossRef]
64. Parra, C.; Blanc, X.; Cleve, A.; Duchien, L. Unifying design and runtime software adaptation using aspect models. *Sci. Comput. Program.* **2011**, *76*, 1247–1260. [CrossRef]
65. Olaechea, R.; Atlee, J.; Legay, A.; Fahrenberg, U. Trace checking for dynamic software product lines. In Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems, Gothenburg, Sweden, 28–29 May 2018; pp. 69–75.
66. Cubo, J.; Gamez, N.; Pimentel, E.; Fuentes, L. Reconfiguration of service failures in damasco using dynamic software product lines. In Proceedings of the 2015 IEEE International Conference on Services Computing, New York, NY, USA, 27 June–2 July 2015; pp. 114–121.
67. Nakanishi, T.; Furusho, H.; Hisazumi, K.; Fukuda, A. Dynamic SPL and derivative development with uncertainty management for DevOps. In Proceedings of the 2016 5th IEEE IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), Kumamoto, Japan, 10–14 July 2016; pp. 244–249.

68. Shen, L.; Peng, X.; Zhao, W. Software Product Line Engineering for Developing Self-Adaptive Systems: Towards the Domain Requirements. In Proceedings of the 2012 IEEE 36th Annual Computer Software and Applications Conference, Izmir, Turkey, 16–20 July 2012; pp. 289–296. [CrossRef]
69. Nascimento, A.S.; Rubira, C.M.; Castor, F. ArCMAPE: A Software Product Line Infrastructure to Support Fault-Tolerant Composite Services. In Proceedings of the 2014 IEEE 15th International Symposium on High-Assurance Systems Engineering, Miami Beach, FL, USA, 9–11 January 2014; pp. 41–48. [CrossRef]
70. Metzger, A.; Bayer, A.; Doyle, D.; Sharifloo, A.M.; Pohl, K.; Wessling, F. Coordinated run-time adaptation of variability-intensive systems: An application in cloud computing. In Proceedings of the 2016 IEEE/ACM 1st International Workshop on Variability and Complexity in Software Design (VACE), Austin, TX, USA, 15 May 2016; pp. 5–11.
71. Lochau, M.; Bürdek, J.; Hölzle, S.; Schürr, A. Specification and automated validation of staged reconfiguration processes for dynamic software product lines. *Softw. Syst. Model.* **2017**, *16*, 125–152. [CrossRef]
72. Gamez, N.; Romero, D.; Fuentes, L.; Rouvoy, R.; Duchien, L. Constraint-based self-adaptation of wireless sensor networks. In Proceedings of the 2nd International Workshop on Adaptive Services for the Future Internet and 6th International Workshop on Web APIs and Service Mashups, Bertinoro, Italy, 19 September 2012; pp. 20–27.
73. de Sousa Santos, I.; de Jesus Souza, M.L.; Luciano Carvalho, M.L.; Alves Oliveira, T.; de Almeida, E.S.; de Castro Andrade, R.M. Dynamically Adaptable Software Is All about Modeling Contextual Variability and Avoiding Failures. *IEEE Softw.* **2017**, *34*, 72–77. [CrossRef]
74. Pfannemuller, M.; Krupitzer, C.; Weckesser, M.; Becker, C. A Dynamic Software Product Line Approach for Adaptation Planning in Autonomic Computing Systems. In Proceedings of the 2017 IEEE International Conference on Autonomic Computing (ICAC), Columbus, Ohio, USA, 17–21 July 2017; pp. 247–254. [CrossRef]
75. Göttmann, H.; Bacher, I.; Gottwald, N.; Lochau, M. Static Analysis Techniques for Efficient Consistency Checking of Real-Time-Aware DSPL Specifications. In Proceedings of the 15th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS'21), Krems, Austria, 9–11 February 2021; Association for Computing Machinery: New York, NY, USA, 2021. [CrossRef]
76. Marinho, F.G.; Maia, P.H.; Andrade, R.M.; Vidal, V.M.; Costa, P.A.; Werner, C. Safe adaptation in context-aware feature models. In Proceedings of the 4th International Workshop on Feature-Oriented Software Development, Dresden, Germany, 24–25 September 2012; pp. 54–61.
77. Murguzur, A.; Capilla, R.; Trujillo, S.; Ortiz, Ó.; Lopez-Herrejon, R.E. Context variability modeling for runtime configuration of service-based dynamic software product lines. In Proceedings of the 18th International Software Product Line Conference: Companion Volume for Workshops, Demonstrations and Tools, Florence, Italy, 15–19 September 2014; Volume 2, pp. 2–9.
78. Baresi, L.; Quinton, C. Dynamically evolving the structural variability of dynamic software product lines. In Proceedings of the 2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, Florence, Italy, 18–19 May 2015; pp. 57–63.
79. Pessoa, L.; Fernandes, P.; Castro, T.; Alves, V.; Rodrigues, G.N.; Carvalho, H. Building reliable and maintainable dynamic software product lines: An investigation in the body sensor network domain. *Inf. Softw. Technol.* **2017**, *86*, 54–70. [CrossRef]
80. Achtaich, A.; Roudies, O.; Souissi, N.; Salinesi, C.; Mazo, R. Evaluation of the State-Constraint Transition Modelling Language: A Goal Question Metric Approach. In Proceedings of the 23rd International Systems and Software Product Line Conference, Paris, France, 9–13 September 2019; Volume B, pp. 106–113.
81. Frantzi, K.; Ananiadou, S.; Mima, H. Automatic recognition of multi-word terms: The c-value/nc-value method. *Int. J. Digit. Libr.* **2000**, *3*, 115–130. [CrossRef]
82. Petersen, K.; Gencel, C. Worldviews, Research Methods, and their Relationship to Validity in Empirical Software Engineering Research. In Proceedings of the 2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement, Ankara, Turkey, 23–26 October 2013; pp. 81–89. [CrossRef]
83. Alférez, G.H.; Pelechano, V. Achieving autonomic Web service compositions with models at runtime. *Comput. Electr. Eng.* **2017**, *63*, 332–352. [CrossRef]
84. Bezerra, C.; Lima, R.; Silva, P. DyMMer 2.0: A Tool for Dynamic Modeling and Evaluation of Feature Model. In *Brazilian Symposium on Software Engineering*; Association for Computing Machinery: New York, NY, USA, 2021; pp. 121–126.