A Customizable SPL Scoping Process for SPL Reengineering

Luciano Marchezan[1], **Elder Rodrigues**[1], **Maicon Bernardino**[1],
Fábio Paulo Basso[1]

[1]Universidade Federal do Pampa (UNIPAMPA)
Av. Tiarajú, 810, Ibirapuitã – Alegrete, RS – Brasil

{lucianomarchezan94, fabiopbasso}@gmail.com

elderrodrigues@unipampa.edu.br, bernardino@acm.org

Abstract. *Software Product Lines (SPL) are a well known solution to systematically create reusable software products. SPL reengineering emerges as a strategy for obtaining SPL from a set of legacy systems. As there are different scenario variables, such as, available artifacts and team experience, the activities and techniques used to perform reengineering tasks may change, requiring that approaches provide some customization to user's scenarios. In this context, SPL scoping is important for defining domain variables used for decisions, intending to improve the cost benefit of SPL. However, there is a lack of a process supporting these tasks considering different scenarios. Therefore, we specify Prepare, Assemble and Execute Process for SPL Reengineering (PAxSPL), a process that provides support to prepare, assemble and execute feature retrieval while also considering SPL scoping concepts and activities. In this work, we present our approach and perform a comparison to related works showing that PAxSPL customization may reduce effort of adoption in user's scenarios.*

1. Introduction

Software Product Lines (SPL) is a common practice in organizations that aim at reusing software products systematically [Pohl et al. 2005]. To achieve this goal, the SPL engineering emerged with a process life-cycle divided in domain engineering and application engineering [Van der Linden et al. 2007]. Both phases of the life cycle share similar activities, such as requirements engineering. Activities, however, are differentiated among the phases with regard to their generated artifacts. While domain engineering activities focus on more high-level artifacts through fragmentation techniques, application engineering generates more concrete through composition and merge techniques, i.e. generating low-level artifacts. There is, however, an activity exclusively performed only during the domain engineering phase: product management. This activity aims to define the scope of the SPL and generates a product roadmap, defining the common and variable features of the products [Kang et al. 2002].

The product management sub-process, also know as SPL Scoping [de Moraes et al. 2009], is devoted to analyze economic opportunities promoted by the products [John and Eisenbarth 2009]. Likewise, SPL scoping is responsible for identifying features that will give an optimal return of investment in time to market requirements [Schmid 2002]. However, as organizational scenarios may vary, the SPL scoping proposals should be customizable according to a given scenario. In addition, as the SPL extractive approach is widely used in the industry [Krueger 2001], SPL scoping approaches should consider the adoption to SPL reengineering scenarios. However, only few SPL scoping approaches were designed for SPL reengineering [Noor et al. 2008]. Therefore, by evolving previous work [Marchezan et al. 2019], we intend to provide a customizable process that considers SPL Scoping in reengineering scenarios. Our proposal, PAxSPL, provides guidelines for users define their own customized scoping process which is performed in parallel to feature retrieval and analysis, common activities of SPL reengineering.

The remainder of this work is structured as follows: Section 2 presents the main concepts of our research; Section 3 presents our approach and its guidelines; Section 4 compares related works with PAxSPL; Finally, Section 5 presents our conclusions.

2. Background

2.1. Software Product Line Engineering

SPL is defined by [Clements and Northrop 2001] as a set of software-intensive systems that share a common and managed set of features satisfying the specific needs of a particular market and that are developed from a common set of core assets in a prescribed way. As defined by [Pohl et al. 2005], "SPLE is the paradigm responsible for the development and study of SPLs. It uses platforms and mass customization concepts to enable variability management".

[Pohl et al. 2005] also proposed a model for SPLE which is divided between domain engineering and application engineering. Each activity of domain engineering generates a high-level artifact that will be used in application engineering. These high-level artifacts are: reference requirements, reference architecture, domain realization mechanisms, and domain tests. They are later specialized in application artifacts: software requirements specifications, software architecture, source code, and test cases.

There are three possible approaches for an organization moving from traditional software to SPL: proactive, reactive, and extractive [Krueger 2001]. By using the proactive approach, an organization would plan, analyze, design and develop a complete SPL including the whole scope of their products. With the reactive approach, the organization would increase their SPL development to reach the demand for new products, or emerging new requirements. The extractive approach, however, is used when an organization already has all its products developed in a non-systematic manner. The extraction of common and varying source code is performed and their products are transformed into an SPL. However, an extractive approach is the most indicated when the organization already has a set of software variants because it allows to identify and extract the commonalities and the variabilities among them. This process of extraction is called SPL Reengineering [Assunção et al. 2017].

2.2. Software Product Line Reengineering

The term reengineering can be described as "the examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form" [Chikofsky and Cross 1990]. In the SPL context, reengineering is used to transform a system, system family, or system variants into an SPL. According to [Assunção et al. 2017], the SPL reengineering process is composed of three main phases: detection, analysis and transformation. During the first phase, detection, variability and commonalities of the products are identified and extracted from the system variants through the use of feature retrieval techniques. These characteristics are represented in the form of features.

Techniques and methods used during this phase aim to extract data from artifacts, such as class diagrams and source code. The second phase is analysis, where the discovered features are organized as a feature model. Feature model is the mostly used mechanism to represent SPL variability [Clements and Northrop 2001]. To the best of our knowledge, the first feature model notation, FODA, was presented by [Kang et al. 1990]. In the SPL context, the feature model is represented in a tree structure, where its root is usually the SPL being modeled. The last phase, transformation, is when artifacts linked with these features are managed and modified to create the SPL. During the first two phases of this process, feature retrieval techniques are used to retrieve the variabilities and commonalities of the products.

2.3. SPL Scoping

A planning to apply SPL-based techniques considering business alignment and goals is called as scoping [Jhon 2010]. Literature abounds with proposals for SPL-based production plans as surveyed in [John and Eisenbarth 2009]. These works are devoted to characterize particular artifacts and tasks for specific domains. Proposals such as PuLSE are well-known in the literature and despite being developed for being customizable, the approach does not provide guidance for adaptability for a reengineering context.

3. SPL Scoping for SPL Re-engineering

In this section we describe how PAxSPL customization is possible. We also explain how we handle SPL scoping in the SPL reengineering context.

3.1. Customization for Different Scenarios

The Prepare, Assemble and Execute Process for SPL Reengineering (PAxSPL) was designed for giving its users enough guidance when conducting feature retrieval. As organizational contexts change, the approach must cover these changes. Thus, we defined guidelines with alternatives techniques and strategies for performing the retrieval. As illustrated in Figure 1, we grouped the techniques based on their strategy. We have the mandatory group of Retrieval Techniques which are composed by two Or-alternatives (at least one must be selected [Czarnecki and Eisenecker 1999]), Static Analysis and Information Retrieval techniques.

For Static analysis we have three Or-alternatives: Dependency Analysis and its variations, Data-Flow analysis and its variations, and Clustering. For information retrieval we also have three Or-alternatives: Latent Semantic Indexing (LSI), Vector Space Model (VSM) [Alves et al. 2008] and Formal Concept Analysis (FCA). The second group is optional, composed by three techniques: Expert Driven Extraction, Rule-Based Techniques, and Heuristics. Based on the selection of the techniques, the user would assemble them into the generic process for feature retrieval, showed in Figure 2. The process is composed by four basic activities. An example of assembled process would be using FCA during the *Extract* task, LSI for the *Categorize* and clustering for *Group*. Then, the feature artifacts would be converted to a feature model.

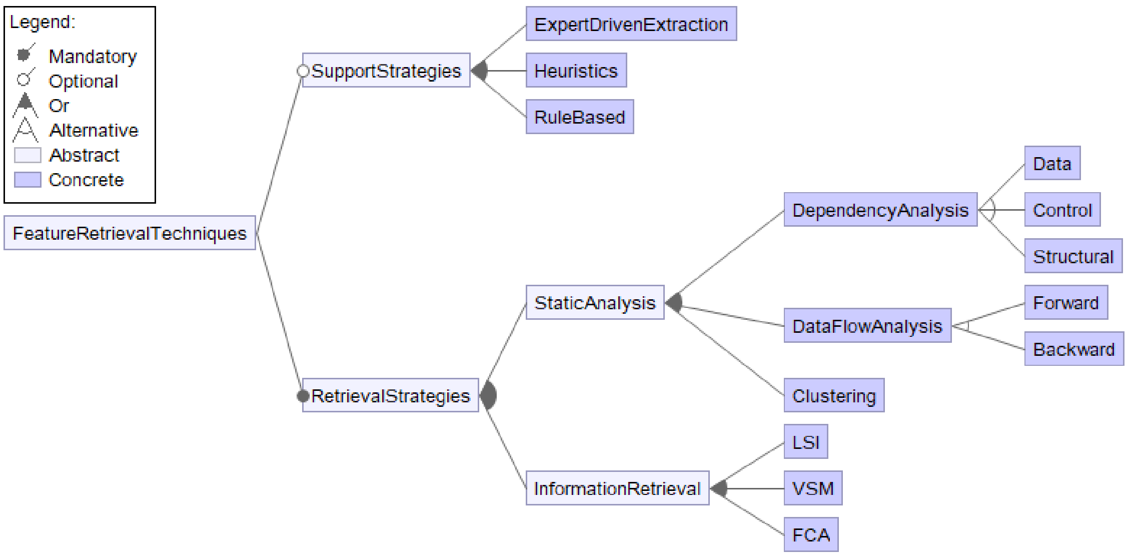

Figure 1. A Feature Model of Retrieval Techniques.

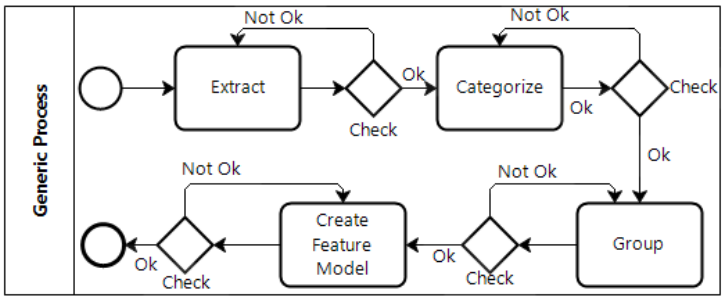

Figure 2. The Generic Process for Feature Retrieval.

In addition to the feature retrieval, PAxSPL also provides guidance for customization considering the Scope of the SPL. By analyzing a set of 38 works citing SPL scoping proposals, we were also able to establish a feature model of Scoping activities and concepts. Figure 3 presents these activities which are divided by Scoping type and Supporting concepts.

All features in the model are Or-alternative, except by the supporting concepts, which are optional. Among the supporting, we have the definition of metrics for scoping, use or definition of meta-models and the SPL evolution plan. Considering the scoping types, they are three: Domain, Asset and Product. Domain is subdivided in market analysis, cost benefit analysis and product roadmap. Continuing, the asset scoping may be divided into prioritize products, definition of architecture and variability analysis. Lastly, product scoping, also called product portfolio, is composed of candidate analysis and feature definition.

Similar to the feature retrieval strategies, the scoping activities and concepts must be selected by PAxSPL's users and assembled into a generic SPL Scoping process, presented in Figure 4. *Pre-Scoping* is the first task, where supporting concepts from the Scoping feature model may be used, such as metrics definition.

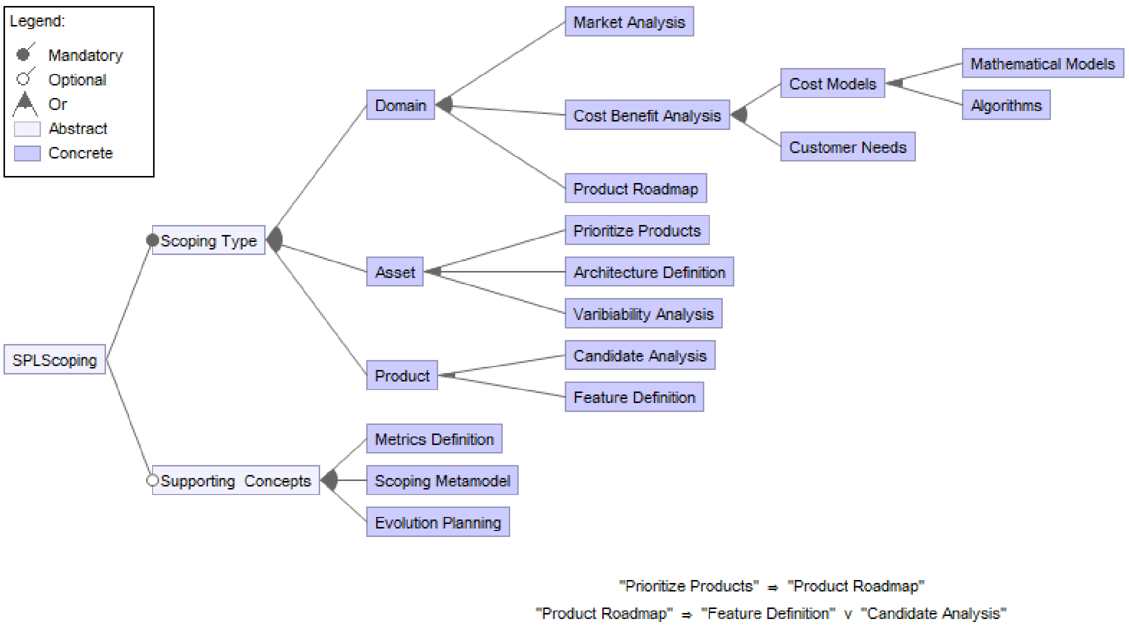

Figure 3. A feature Diagram of SPL Scoping Activities

Then, the users would assemble the activities related which scoping type. The selection of activities is performed based on the user's context. For instance, a market analysis may be performed in the *domain scoping* activity. As presented in the BPMN model, the activities related with the scoping type are not mandatory, however, at least one must be performed. Lastly, the *scoping closure* activity is executed. This activity is generic as several ways to close the scoping process may be performed, such as the evolution planing.

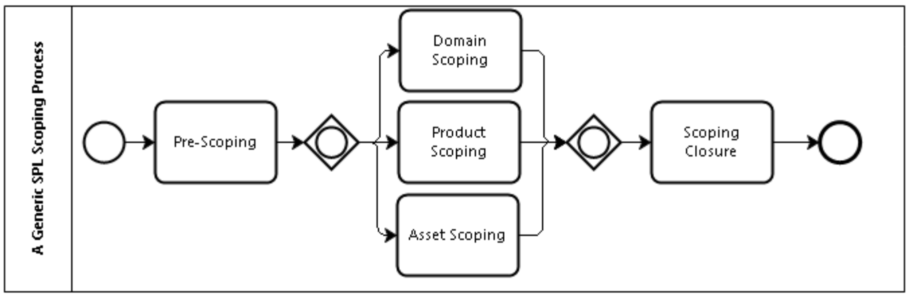

Figure 4. A Generic SPL Scoping Process.

3.2. PAxSPL Overview

To aid the decisions related with selecting the strategies and techniques for feature retrieval, and SPL scoping, we defined PAxSPL main workflow. The process is presented in Figure 5, divided in three main phases: prepare, assemble and execute.

Prepare: during this phase, team information is collected including experience, skills, knowledge and preferences of each member. Then, roles are assigned based on the team information collected.

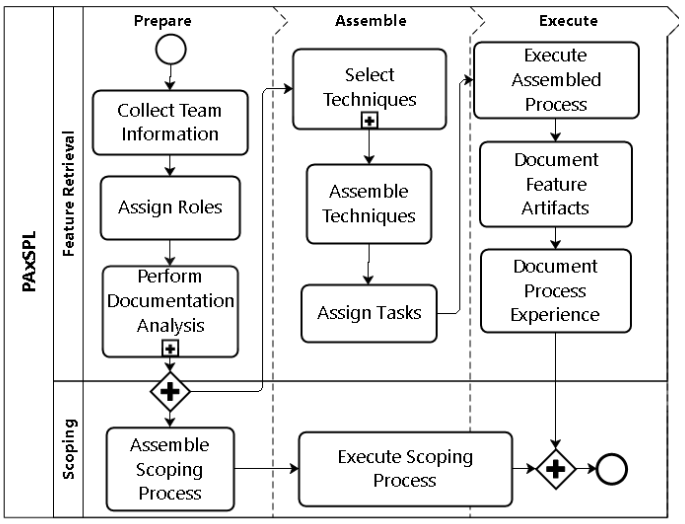

Figure 5. The Prepare, Assemble and Execute Process for Software Product Line Reengineering.

Possible roles are: Domain Engineer, Analyst, Architect and Developer. These roles are related with the following sub-process, which is *Perform Documentation Analysis*. Here, domain information, constraints, requirements information, architecture, technologies and additional information from the systems being analyzed are collected. Some artifacts have a higher level of impact when choosing a technique for retrieving the features, but they all must be used to assemble the process. After this activity, we have a parallel gateway which divides the main workflow into the feature retrieval and scoping. Still during the Prepare phase, the scoping process should be assembled using the scoping feature model and the scoping generic process.

Assemble: in this phase, the data collected previously is analyzed to help the selection of techniques for feature retrieval. The users should analyze whether their scenario contains some similarity in comparison with the scenarios on which the technique was used (such information is present in PAxSPL guidelines). The second activity is *Assemble Techniques* where the chosen techniques are assembled inside our generic process, shown in Figure 2. The last activity is *Assign Tasks*, where each member of the team will receive a task to perform during the retrieval process execution. In this case, one member may perform multiple tasks and a task can be performed by multiple members. In parallel with these activities, the scoping process is being executed.

Execute: during this phase, the feature retrieval is performed and the feature artifacts are collected. The first activity is *Execute Assembled Process*, where the assembled process is executed to detect, extract, categorize, and group the features according to the selected techniques. The second activity is *Document Feature Artifacts*, here, artifacts are documented in a structured way according to the techniques selected. Artifacts may be variability reports, feature descriptions, data dictionary among other. Lastly, reports are created to document the experience of the process execution during the *Document Process Experience* activity. These reports may be used in future re-executions of the process (*e.g.*, when new features emerge from clients demand, or for different software products of the same organization), reducing cost and effort. In parallel, the scoping process is being finished, ending the process execution.

3.3. Guidelines

By analyzing other SPL reengineering approaches, mapping the used strategies and artifacts, we created a set of guidelines. These guidelines contain support documentation to help choosing techniques, describe each one, give examples, supporting tools, define recommended scenarios and give a prioritization assemble order when assembling a technique into the generic process.

Among the many information found in the guidelines section of our process, we included a basic introduction to SPL and its main concepts. We also provided information about variability management and feature model notations and tools. These guidelines are important for selecting both the feature retrieval techniques and scoping concepts. We aimed at giving support to newcomers, however the guidelines may also aid experienced practitioners.

4. Comparison to Related Work

Although we analyzed a set of 38 different studies for creating the guidelines related to SPL scoping, due to space limitation, in this section we only discuss the most cited works.

ASPLE [Abbas and Andersson 2013]: the authors extended the architectural reasoning framework (ARF) to provide models and constructs to domain architects. These models allow to reason about variability in domain quality attributes with self adaptation. The ASPLE framework considers two separate SPL, one for managing subsystems and one for the managed subsystems. These two separate SPL are composed of some activities: i) ASPL Domain engineering: defines and implements a reusable platform for the adaptation logic; ii) Baseline SPL: focuses on application logic.

[Acher et al. 2012]: the approach presented in the study extracts feature models from several tabular data files. This procedure is semi-automated and the features are hierarchically organized. The authors also proposed a language that supports scoping activities which is used to parameterize the features extraction. The scoping is only covered as this single task. A practitioner may scope the data in various ways and for many purposes using the proposed language.

RiPLE-SC [Balbino et al. 2011]: the main focus is to operate in small to medium-sized companies. The general process is divided in four phases: i) pre-scoping: when meetings are scheduled with different project stakeholders to evaluate the SPL availability, benefits, drawbacks, and their market; ii) domain scoping: when the domains and sub-domains are identified and prioritized; iii) product scoping: performed to identify and review features, identify products, and construct and validate a product map; iv) asset scoping: metrics are created and applied to prioritized the features on the product map.

PuLSE-Eco [Schmid 2002]: developed as a customizable method to support the conception, contruction, usage, and evolution of SPL, PuLSE is divided in components. One of this components was created specifically for SPL scoping, the PuLSE-Eco. This component aims at identify the characteristics that are directly supported by the referential architecture. For achieving this, product candidates must be mapped, evaluation functions are developed, products are characterized, a benefit analysis is performed and the SPL plan is developed. By performing these tasks, PuLSE captures on an abstract level the relationships between scoping information and implementation aspects and thus allows to provide rough guidance on implementation aspects of the project.

[Noor et al. 2008]: the process presents guidelines and uses thinkLets for collaboration engineering. The activities are divided among five parts: i) Discuss and agree on domains: participants brainstorm on selected issues related with domains while stakeholders develop a shared understanding of the domains; ii) Assign features to domains: features already known are categorized in appropriate domains, new features are brainstormed and evaluated in quick sessions, participants go through features to ensure that they are categorized appropriately; iii) Agree on products: participants give their opinion on proposed products, to consolidate their vision regarding them; iv) Develop product map: voting is performed to formulate a product map, reason for disagreements are elicited, and a consensus is build upon it; vi) prioritize product map: voting is performed to ascertain the priority of products and features of the product map, reasons for disagreements are revealed, and consensus is built.

CoMeS [Ojeda et al. 2018]: a collaborative way of guiding the definition of the scope, presents the tasks and details the steps with the objective that the stakeholders team could performs the outcomes of each task and at the end of the scoping activity, obtain a represented scope. The activities present in CoMeS are: i) Initial Meeting; ii) Explore existing products; iii) Identify features, products and sub-domains; iv) Specify product map and establish objectives; v) Quantify product map and domains; vi) Closure meeting.

PLEvo-Scping [Villela et al. 2010]: complements and extends SPL scoping approaches by helping the SPL scoping team anticipate emergent features and distinguish unstable from stable SPL features. It contains several activities: i) Preparation for volatility analysis: establishes the basis for the volatility analysis; ii) Environment change anticipation: identifying and characterizing facts that may take place in the SPL environment within the pre-established timeframe, and may allow or require adaptations in the SPL; iii) Change impact analysis: analyze the impact of the identified facts on the SPL; iv) SPL Evolution Planning: establishes when and how relevant adaptation needs are expected to be introduced into the SPL, and prepare it for accommodating the adaptation needs beforehand.

Table 1 presents the comparison among the presented works with PAxSPL. Considering the scoping type, PAxSPL, RiPLE-SC, CoMeS, and PLEvo-Scoping cover all types. Guidelines are present in [Noor et al. 2008], CoMeS, PLEvo-Scoping and PAxSPL as well. The customization considering different scenarios and contexts is given by PuLSE-Eco and PAxSPL. Lastly, the domains are mostly any SPL, with ASPLE focusing on self-adaptive systems, RiPLE-SC in agile methods. PAxSPL and [Noor et al. 2008] are the only two approaches focusing on the SPL reenginering domain.

5. Conclusion

As important as SPL scoping is in a SPL life-cycle, SPL scoping approaches should consider different scenarios and handle adaptation. In addition, approaches considering both scoping and SPL re-engineering are important as the SPL extractive approach is widely used in organizations.

Table 1. Studies Comparison

Study	Scoping Type	Guidelines	Custom.	Domain
ASPLE	D, P	No	No	Self-adaptive systems
Acher *et al.*	D	No	No	SPL
RiPLE-SC	A, D, P	No	No	Agile Methods
PuLSE-Eco	A, D	No	Yes	SPL
Noor *et al.*	A, D	Yes	No	SPL reengineering
CoMeS	A, D, P	Yes	No	SPL
PLEvo-Scoping	A, D, P	Yes	No	SPL
PAxSPL	A, D, P	Yes	Yes	SPL reengineering

D - Domain; P - Product; A - Asset.

We identified a lack in the literature of approaches covering all these topics. Thus, by evolving previous work, we proposed PAxSPL, a feature retrieval process for SPL reengineering that handles SPL scoping activities.

We established a feature model of scoping concepts and activities extracted from a set of 38 approaches found in the literature. Therefore, we were able to create guidelines for users assembling their own customized SPL scoping process. We presented a manual comparison of PAxSPL with some related works, showing that our proposition gives support for all scoping types, customization, provides guidelines and it is designed focused on SPL reengineering. All these characteristics were not found in the other proposals. We aim at improving PAxSPL by planing several evaluations in the future such as a case study in a real organization.

References

Abbas, N. and Andersson, J. (2013). Architectural reasoning for dynamic software product lines. In *Proceedings of the 17th International Software Product Line Conference Co-located Workshops*, SPLC '13 Workshops, pages 117–124, New York, NY, USA. ACM.

Acher, M., Cleve, A., Perrouin, G., Heymans, P., Vanbeneden, C., Collet, P., and Lahire, P. (2012). On extracting feature models from product descriptions. In *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*, VaMoS '12, pages 45–54, New York, NY, USA. ACM.

Alves, V., Schwanninger, C., Barbosa, L., Rashid, A., Sawyer, P., Rayson, P., Pohl, C., and Rummler, A. (2008). An exploratory study of information retrieval techniques in domain analysis. In *2008 12th International Software Product Line Conference*, pages 67–76.

Assunção, W., Lopez-Herrejon, R., Linsbauer, L., Vergilio, S., and Egyed, A. (2017). Reengineering legacy applications into software product lines: a systematic mapping. *Empirical Software Engineering*, pages 1–45.

Balbino, M., Almeida, E., and Meira, S. (2011). An agile scoping process for software product lines. In *Proceedings of SEKE*, pages 717–722.

Chikofsky, E. and Cross, J. (1990). Reverse engineering and design recovery: A taxonomy. *IEEE Software*, 7(1):13–17.

Clements, P. and Northrop, L. (2001). *Software product lines: Practices and Patterns*. Addison-Wesley Professional, 3rd edition.

Czarnecki, K. and Eisenecker, U. (1999). Components and generative programming. In *ACM SIGSOFT Software Engineering Notes*, volume 24, pages 2–19. Springer-Verlag.

de Moraes, M. B. S., de Almeida, E. S., and Romero, S. (2009). A systematic review on software product lines scoping. In *Proceedings of 6th Experimental Software Engineering Latin American Workshop (ESELAW 2009)*, page 63. Citeseer.

Jhon, I. (2010). Using documentation for product line scoping. *Software, IEEE*, 27(3):42–47.

John, I. and Eisenbarth, M. (2009). A decade of scoping: A survey. In *Proceedings of the 13th International Software Product Line Conference*, SPLC '09, pages 31–40.

Kang, K., Cohen, S., Hess, J. A., Novak, W., and Peterson, A. (1990). Feature-oriented domain analysis (FODA) feasibility study. Technical report, DTIC Document.

Kang, K. C., Jaejoon Lee, and Donohoe, P. (2002). Feature-oriented product line engineering. *IEEE Software*, 19(4):58–65.

Krueger, C. (2001). Easing the transition to software mass customization. In *International Workshop on Software Product-Family Engineering*, pages 282–293. Springer.

Marchezan, L., Macedo Rodrigues, E., Bernardino, M., and Paulo Basso, F. (2019). Paxspl: A feature retrieval process for software product line reengineering. *Software: Practice and Experience*, 49(8):1278–1306.

Noor, M. A., Grünbacher, P., and Hoyer, C. (2008). A collaborative method for reuse potential assessment in reengineering-based product line adoption. In Meyer, B., Nawrocki, J. R., and Walter, B., editors, *Balancing Agility and Formalism in Software Engineering*, pages 69–83, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ojeda, M. C. C., Alegria, J. A. H., Rodriguez, F. J. A., and Melenje, P. H. R. (2018). A collaborative method for a tangible software product line scoping. In *2018 ICAI Workshops (ICAIW)*, pages 1–6.

Pohl, K., Böckle, G., and van Der Linden, F. (2005). *Software product line engineering: foundations, principles and techniques*. Springer Science & Business Media.

Schmid, K. (2002). A comprehensive product line scoping approach and its validation. In *Proceedings of the 24th International Conference on Software Engineering*, ICSE '02, pages 593–603, New York, NY, USA. ACM.

Van der Linden, F., Schmid, K., and Rommes, E. (2007). *Software product lines in action: the best industrial practice in product line engineering*. Springer Science & Business Media, 2007th edition.

Villela, K., Dörr, J., and John, I. (2010). Evaluation of a method for proactively managing the evolving scope of a software product line. In Wieringa, R. and Persson, A., editors, *Requirements Engineering: Foundation for Software Quality*, pages 113–127, Berlin, Heidelberg. Springer Berlin Heidelberg.