

UNIVERSIDADE FEDERAL DO PAMPA

Luciano Marchezan

**PAxSPL: A Generic Framework to Support the Planning of SPL
Reengineering**

Alegrete
2020

Luciano Marchezan

**PAXSPL: A Generic Framework to Support the Planning of SPL
Reengineering**

Master thesis presented as partial requirement for obtaining the degree of Masters of Software Engineering at Universidade Federal do Pampa.

Supervisor: Prof. PhD. Elder de Macedo Rodrigues

Co-supervisor: Prof. PhD. Maicon Bernardino

Alegrete
2020

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

M317p Marchezan de Paula, Luciano Augusto
PAxSPL: A Generic Framework to Support the Planning of SPL
Reengineering / Luciano Augusto Marchezan de Paula.
137 p.

Dissertação(Mestrado)-- Universidade Federal do Pampa,
MESTRADO EM ENGENHARIA DE SOFTWARE, 2020.
"Orientação: Elder Rodrigues".

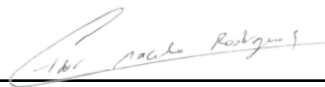
1. Software reuse. 2. Variability Management. 3. Software
Product Lines. 4. Software Reuse. I. Título.

Luciano Marchezan

**PAXSPL: A Generic Framework to Support the Planning of SPL
Reengineering**

Master thesis presented as partial requirement for obtaining the degree of Masters of Software Engineering at Universidade Federal do Pampa.

Master thesis held and approved in ..13. Agosto... of .2020...
Board of examiners:



Prof. PhD. Elder de Macedo Rodrigues

Advisor
Unipampa



Prof. PhD. Maicon Bernardino

Co-Advisor
Unipampa



Prof. PhD. Wesley Klewerton Guez Assunção

UTFPR



Prof. PhD. Fábio Paulo Basso

Unipampa

“I may not have gone where I intended to go,
but I think I have ended up where I needed to be.”
(Douglas Adams)

ABSTRACT

Software Product Line (SPL) is a well-known approach for systematically creating reusable software assets and customized software products for a specific domain or market segment. Among the different approaches for adopting SPL, the extractive approach is a promising solution for organizations that intend to transform their legacy applications into an SPL. In this sense, the SPL reengineering process emerges as a possible strategy for applying the SPL extractive approach. Another important concept related to SPL development is scoping. The SPL scoping process is conducted for defining the boundaries of an SPL, being one of the core planning activities performed during SPL development. Although not deeply discussed in technical SPL contributions, activities to support the definition of SPL development budgets and cost estimations are essential. There are several studies proposed for handling SPL reengineering such as processes, tools, frameworks, meta-models among others. Due to the diversity of options found in daily practice of software development, a rigorous analysis of contexts is critical to instantiate these proposals. As there are different scenario variables, such as available artifacts and team experience, the activities, and techniques used to perform SPL reengineering or SPL scoping tasks may change, requiring from software engineers to adapt their approaches as a means to satisfying the companies' scenarios. To the best of our knowledge, however, there is a lack of an approach supporting these tasks considering different scenarios. Therefore, in this work we propose the Prepare, Assemble, and Execute Framework SPL reengineering (PAxSPL). PAxSPL is composed of three different aspects: a process, guidelines, and a supporting tool. The process provides support to prepare, assemble, and execute feature retrieval and analysis alongside activities considering SPL scoping concepts. Thus, users may plan the scope of their SPL while conducting the retrieval and analysis of features by applying retrieval techniques. The guidelines were created to aid users with low experience in SPL reengineering or SPL scoping. The supporting tool was developed to reduce the effort of managing and executing an SPL reengineering process. This effort reduction is done by automatizing the management and generation of reports. For evaluating PAxSPL customization capabilities, we extracted eight different scenarios from the literature used as input for application of PAxSPL. The results evidenced that PAxSPL is customizable to a variety of scenarios with different artifacts, retrieval techniques, and activities. However, we also identified some challenges that limited the customization level of our framework. Based on these challenges, we perform improvements to increase the PAxSPL customization level.

Key-words: Software Reuse, Variability Management, Software Product Lines, Software reengineering.

RESUMO

Linhas de produtos de *software* (LPS) é uma abordagem bem conhecida para se criar de maneira sistemática *assets* de *software* reusáveis além de produtos de *software* customizados para um domínio específico. Dentre as diferentes abordagens para se adotar LPS, a abordagem extrativa é uma solução promissora em organizações que pretendem transformar seus sistemas legados em LPS. Nesse contexto, o processo de reengenharia para LPS surge como uma possível estratégia para se aplicar a abordagem extrativa de LPS. Outro conceito importante relacionado com LPS é o escopo. O processo de escopo de LPS é executado para se definir os limites de uma LPS, sendo uma das atividades essenciais de planejamento executadas durante o desenvolvimento de LPS. Apesar de não ser discutido de maneira aprofundada em contribuições técnicas, as atividades que dão suporte a definição dos orçamentos e estimação de custos de desenvolvimento são essenciais. Para ambos os tópicos, reengenharia e escopo de SPL, existem diversos trabalhos propostos, como processos, ferramentas, *frameworks*, meta-modelos entre outros. Por conta da diversidade de opções encontradas em práticas diárias de desenvolvimento de software, uma análise rigorosa de contextos é crucial para instanciar estas propostas. Como existem diferentes variações de cenário, como artefatos disponíveis e a experiência do time, atividades e técnicas utilizadas para se executar a reengenharia ou o escopo de LPS podem variar, necessitando que abordagens se adaptem buscando satisfazer os cenários dos usuários. Porém, identificamos a falta de uma abordagem que dá suporte a essas atividades considerando diferentes cenários. Portanto, neste trabalho propomos o *Prepare, Assemble and Execute Framework for SPL Reengineering* (PAXSPL). PAXSPL é composto por três aspectos distintos: um processo, diretrizes e uma ferramenta de suporte. O processo fornece suporte para se preparar, montar e executar recuperação de *features* considerando conceitos e atividades de escopo. Portanto, os usuários podem planejar o escopo da LPS enquanto conduzem a recuperação e análise de *features* aplicando as técnicas de recuperação. As diretrizes foram criadas com a intenção de auxiliar usuários com pouca experiência em reengenharia ou escopo de LPS. A ferramenta de suporte foi desenvolvida com o objetivo de reduzir o esforço de gerenciamento e execução do processo de reengenharia, alcançada através da automatização do gerenciamento e geração de relatórios. Para avaliar a capacidade de customização do PAXSPL, extraímos oito cenários da literatura, utilizados como entrada para a aplicação do PAXSPL. Os resultados evidenciaram que PAXSPL é customizável para uma variedade de cenários com diferentes artefatos, técnicas de recuperação e atividades. Porém, também identificamos oito desafios que limitaram o nível de customização de nosso *framework*. Baseado nestes, aplicamos melhorias para aumentar o nível de customização do PAXSPL.

Palavras-chave: Reuso de Software, Gerenciamento de Variabilidade, Linhas de Produto de Software, Reengenharia de Software.

LIST OF FIGURES

Figure 1 – The two-life-cycle model of SPLE.	26
Figure 2 – SPL Reengineering process.	29
Figure 3 – A feature model of a car.	31
Figure 4 – A feature model created using the Generative Programming notation.	32
Figure 5 – Research Methodology Design	35
Figure 6 – SPL Scoping Concept Map	46
Figure 7 – A Generic SPL Scoping Process	54
Figure 8 – Evaluations by Year	59
Figure 9 – PAXSPL Framework	69
Figure 10 – The Prepare, Assemble and Execute Process for Software Product Line Reengineering.	70
Figure 11 – Perform Documentation Analysis Sub-process.	71
Figure 12 – Select Techniques Sub-process.	72
Figure 13 – The Generic Process for Feature Retrieval and Analysis.	72
Figure 14 – PAXSPL Execute Phase	73
Figure 15 – A Feature Model of Retrieval Techniques.	74
Figure 16 – An Assembled Process	75
Figure 17 – A feature Diagram of SPL Scoping Activities	75
Figure 18 – A Generic SPL Scoping Process.	76
Figure 19 – An Assembled SPL Scoping Process.	77
Figure 20 – PAXSPL Supporting Tool Class Diagram.	81
Figure 21 – Project Home Screen	83
Figure 22 – Artifacts Screen	83
Figure 23 – Techniques Screen	84
Figure 24 – Activities Screen	85
Figure 25 – BPMN Modeling Screen	85
Figure 26 – Execute Activities Screen	86
Figure 27 – Feature Model Screen	87
Figure 28 – Feature Report	88
Figure 29 – Traceability Matrix Screen	88
Figure 30 – Feature Configurator Screen	89
Figure 31 – Product Traceability Matrix Screen	90
Figure 32 – Techniques Report for (EYAL-SALMAN; SERIAI; DONY, 2013)	95
Figure 33 – BPMN Process Assembled for (EYAL-SALMAN; SERIAI; DONY, 2013)	96
Figure 34 – Part of the Process Assembled Report for (EYAL-SALMAN; SERIAI; DONY, 2013)	96
Figure 35 – BPMN Process Assembled for (ACHER et al., 2013)	97

Figure 36 – Process Assembled Report for (ACHER et al., 2013)	98
Figure 37 – BPMN Process Assembled for (AL-MSIE'DEEN et al., 2012)	98
Figure 38 – Process Assembled Report for (AL-MSIE'DEEN et al., 2012)	99
Figure 39 – BPMN Process Assembled for (SHATNAWI; SERIAI; SAHRAOUI, 2014)	100
Figure 40 – BPMN Process Assembled for (ALVES et al., 2008)	100
Figure 41 – BPMN Process Assembled for (CHEN et al., 2005)	101
Figure 42 – BPMN Process Assembled for (PAŠKEVIČIUS et al., 2012)	101
Figure 43 – BPMN Process Assembled for (BREIVOLD; LARSSON; LAND, 2008)	102
Figure 44 – New Generic Process for Feature Retrieval and Analysis.	106
Figure 45 – Changes Applied to the Generic Process for Feature Retrieval and Anal- ysis.	106
Figure 46 – Changes Applied to PAXSPL Process.	107
Figure 47 – Team Information Report Generated by PAXSPL Tool	134
Figure 48 – Artifacts Information Report Generated by PAXSPL Tool	135
Figure 49 – Retrieval Techniques Report Generated by PAXSPL Tool	136

LIST OF TABLES

Table 1 – Terms, Synonyms and the Search String	38
Table 2 – Digital Libraries and Search Strings	39
Table 3 – Inclusion/Exclusion Criteria	39
Table 4 – Numbers of the SLR on Scoping	41
Table 6 – Traceability of Scoping Concepts in the Primary Studies	47
Table 8 – Summary of Related Work	67
Table 9 – Use Cases for PAXSPL Tool	80
Table 11 – Results from the Evaluation.	103
Table 12 – Challenges Identified During the Evaluation.	105
Table 13 – Comparison among related works and PAXSPL (SPL Reengineering). . .	111
Table 14 – Comparison of SPL Scoping Concepts	113

LIST OF ACRONYMS

BPMN Business Process Model and Notation

DD Design Decision

FCA Formal Concept Analysis

FM Feature Model

FODA Feature-Oriented Domain Analysis

LSI Latent Semantic Indexing

PAxSPL Prepare Assemble and Execute Framework for Software Product Line Reengineering

PS Possible Solution

RQ Research Question

SLR Systematic Literature Review

SMS Systematic Mapping Study

SPL Software Product Lines

SPLA Software Product Line Architecture

SPLE Software Product Line Engineering

UC Use Case

UML Unified Modeling Language

VSM Vector Space Model

TABLE OF CONTENTS

1	INTRODUCTION	21
1.1	Context	21
1.2	Motivation	22
1.3	Objectives	23
1.4	Organization	23
2	BACKGROUND	25
2.1	Software Product Line	25
2.1.1	Software Product Line Engineering	26
2.1.1.1	Domain Engineering	26
2.1.1.2	Application Engineering	27
2.1.2	Software Product Line Reengineering	28
2.1.3	Feature Retrieval Techniques	29
2.1.4	Feature Model	30
2.1.5	SPL Scoping	31
2.2	Software Process	32
2.3	Organizational Scenarios	33
2.4	Chapter Lessons	33
3	METHODOLOGY	35
3.1	Establishing Relevance	35
3.2	PAxSPL First Evaluation	36
3.3	Steps for Improvement	36
3.4	New Evaluation	36
4	A SYSTEMATIC LITERATURE REVIEW ON SPL SCOPING	37
4.1	SLR Design and Execution	37
4.2	Results and RQ Answers	41
4.2.1	RQ1. What are the similarities and differences among the approaches?	41
4.2.1.1	Approach Similarities and Differences	45
4.2.1.2	Approach Activities	47
4.2.1.3	Types of Scoping	54
4.2.1.4	Adaptation	57
4.2.2	RQ.2 How are existing scoping approaches evaluated?	58
4.2.2.1	Evaluations Applied	58
4.2.2.2	Evaluation Domains	59

4.2.3	RQ.3 How is the decision making during the process of SPL scope definition?	60
4.2.3.1	Cost Models	60
4.2.3.2	Metrics for Scoping	61
4.2.4	RQ.4. What are the open research gaps and opportunities for new studies on the topic of SPL scoping?	62
4.2.5	Threats to Validity	65
4.3	Related Work	66
4.4	Chapter Lessons	68
5	PAXSPL	69
5.1	PAXSPL Process	69
5.1.1	Prepare	69
5.1.2	Assemble	70
5.1.3	Execute	73
5.2	Customization for Different Scenarios	73
5.2.1	Customization for Feature Retrieval	74
5.2.2	Customization for SPL Scoping	75
5.3	Guidelines	76
5.3.1	Support Checklist	77
5.3.2	Retrieval Techniques Tool Support	79
5.4	PAXSPL Tool	79
5.4.1	Requirements	79
5.4.2	Design	80
5.4.3	Running Example	82
5.5	Chapter Lessons	87
6	EVALUATION	91
6.1	Design	91
6.1.1	Data Set	91
6.1.2	Procedure	92
6.2	Execution	93
6.2.1	Results	94
6.2.2	Discussion	102
6.2.3	How does PAXSPL suit different scenarios?	102
6.2.4	What challenges are observed by customizing PAXSPL?	103
6.3	Improvements	105
6.3.1	Modifications in the Generic Process	105
6.3.2	Modifications in the Guidelines and Documentation	106
6.3.3	Modification in the PAXSPL Process	107

6.4	Chapter Lessons	108
7	RELATED WORK	109
7.1	SPL Reeengineering	109
7.1.1	Studies Main Contribution	109
7.1.2	Artifacts and Strategies for Feature Retrieval	110
7.2	SPL Scoping	112
7.2.1	Guidelines to Support SPL Scoping	112
7.2.2	Customization for Different Scenarios	112
7.2.3	Approaches Domains	113
7.3	Chapter Lessons	114
8	CONCLUSION	115
	BIBLIOGRAPHY	119
	ANNEX A – REPORTS FROM PAXSPL TOOL	133
	Index	137

1 INTRODUCTION

1.1 Context

Increasing the quality of software products is a concern that software development companies have to handle carefully. For software development companies ensure the quality of software products is crucial to guarantee the return of investment. Software development companies, however, also have to balance these investments to reduce project costs. These important aspects have led companies to look for technological solutions that may mitigate these concerns. In this sense, software reuse may be adopted for creating software products from existing systems, thus reusing artifacts, assets, and expertise from the development team. However, there are different strategies for applying software reuse (KRUEGER, 1992), providing companies with different solutions for solving the aforementioned problem.

One possible solution arose with the definition of the Software Product Line Engineering (SPLE) (LINDEN; SCHMID; ROMMES, 2007). Similar to the implementation of the product line in the automobile industry, the introduction of product lines in software production resulted in several changes for the software development industry. As Software Product Lines (SPL) provides a systematic way to reuse software assets when creating new systems, to consider introducing the SPLE for replacing traditional software, development companies have to be aware of SPLE benefits. Among these benefits is the reduction of the maintenance effort and artifacts complexity, as well as the increasing reusability of assets and better cost estimations (POHL; BÖCKLE; LINDEN, 2005).

As the SPLE evolved, different approaches emerged for dealing with SPL development and commercial aspects. The development life-cycle changed when compared with traditional software, and commercial aspects, such as market analysis, were now part of the major software development activities. Among these approaches, Linden, Schmid e Rommes (2007) defined a process life-cycle composed of two phases: domain engineering and application engineering. Both phases of the life cycle share similar activities, such as requirements engineering. These activities, however, are differentiated between the phases concerning their generated artifacts. While domain engineering activities focus on more high-level artifacts by fragmentation techniques, application engineering generates more concrete artifacts through composition and merge techniques, *i.e.*, generating low-level artifacts. While many systems started their development using the two life-cycle approach from scratch, there are also many systems developed as traditional software that requires to be migrated into SPL. In this sense, start from scratch may not be the best solution as many assets of the legacy system might be still useful. These assets include artifacts, domain information, business aspects, and even the developer's knowledge. Hence, how may the companies migrate their traditional or legacy systems into SPL?

A possible answer to the latter question is the SPL reengineering process. This process, which is part of the SPL extractive approach (KRUEGER, 2001), might be used to transform those legacy software products into SPL. This process uses feature retrieval strategies and techniques to identify, extract, and categorized several product variants and transform them into SPL with the use of techniques, methods, and tools (ASSUNÇÃO et al., 2017).

1.2 Motivation

Similar to the SPLE development process proposed by Linden, Schmid e Rommes (2007), different approaches emerged to manage SPL reengineering. As mapped by Assunção et al. (2017), a large number of approaches were proposed along the years to cover different phases of the SPL reengineering process. Studies such as Eyal-Salman, Seriai e Dony (2014) and Al-Msie’Deen et al. (2012) use Formal Concept Analysis (FCA) for retrieving the features of legacy systems while Yu et al. (2013) and Nöbauer, Seyff e Groher (2014) used clustering algorithms. These are only two feature retrieval techniques used for some studies among the many existing ones. As also shown by Assunção et al. (2017), the variety of scenarios and domains require different techniques to be applied. Hence, one approach may not be applicable in different scenarios, limiting its adherence. For instance, an approach that utilizes Unified Modeling Language (UML) diagrams as input for identifying the features of a SPL. In this context, if an organization does not work with UML or if the diagrams are outdated, such an approach would not be useful in this scenario. Thus, to address this problem, the approach should be customizable.

Also, there is the need for formalizing the reengineering process through guidelines as argued by Otsuka et al. (2011) and Ziadi et al. (2012). Also, Kang et al. (2005) advocate that guidelines are needed for evaluating product-line assets, making those assets more reliable. Other authors, such as Martinez et al. (2015) and Stoermer e O’Brien (2001), have pointed out that guidelines may lead to automated support for this process.

To the best of our knowledge, a customizable and well-defined process for performing the SPL reengineering planning, including feature retrieval and analysis, would narrow these gaps and limitations. Thus, in a previous work (MARCHEZAN et al., 2019b), we proposed Prepare and Assembled Process for SPL Reengineering (PAxSPL). PAxSPL is a process supported by guidelines that help users to customize a feature retrieval process for their context. More specifically, PAxSPL gives support to the first two phases of the reengineering process: detection and analysis. In this work, we define these two phases as the “planning” process of SPL reengineering.

After the execution of a case study for evaluating PAxSPL, we identified a few aspects for improvement. Among these, we understand that activities covering SPL scoping were not given the required focus during PAxSPL execution. Such activities would help engineers handle more business-related aspects of the reengineering. The study also

pointed out other pieces of evidence that some PAXSPL activities required high effort due to the need for keeping artifacts and reports updated and well-structured throughout the process execution. This issue implied a need for a supporting tool, aiming at reducing such effort.

1.3 Objectives

Based on the limitations pointed by the studies discussed in the last section, as well as the future work established in our previous work (MARCHEZAN et al., 2019b), we defined the main goal of this work:

- To evolve the PAXSPL process, transforming it into a generic framework to support the planning process of reengineering legacy applications into SPL.

This goal may be divided into different specific objectives:

- i Evolve the PAXSPL process life-cycle, including SPL scoping specific activities;
- ii Include the most common practices of SPL scoping in PAXSPL guidelines, providing a mechanism to define a SPL scoping process to specific contexts;
- iii Develop a supporting tool that may be used for executing all PAXSPL activities;
- iv Conduct an evaluation to collect evidence about PAXSPL customization capabilities.

1.4 Organization

We organized this document as follows:

- **Chapter 2: Background** - Details the main concepts related to our work, such as SPL and feature retrieval strategies;
- **Chapter 3: Methodology** - Explains our research methodology adopted while conducting this project;
- **Chapter 4: SLR on Scoping** - Describes a systematic literature review performed for collecting data from the literature used for expanding our process and guidelines;
- **Chapter 5: PAXSPL** - Description of our proposed process and its guidelines;
- **Chapter 6: Evaluation** - Presents the planned evaluations protocols;
- **Chapter 7: Related Work** - Presents studies similar to PAXSPL, discussing their similarities and differences;
- **Chapter 8: Conclusion** - Presents the preliminary considerations.

2 BACKGROUND

In this chapter, we introduce the terminology and describe the main concepts addressed throughout this work such as SPL in Section 2.1. In Section 2.1.1 the main concepts of SPLE are presented. Section 2.1.2 presents the process of SPL Reengineering. Feature retrieval is discussed in Section 2.1.3. The definition and characteristics of Feature Models are shown in Section 2.1.4. Software processes are discussed in Section 2.2. Lastly, our definition of organizational scenarios is presented and discussed in Section 2.3.

2.1 Software Product Line

The product line concept and its characteristics came from the automobile industry. Henry Ford introduced product lines to car manufacturing in 1913, hoping to meet the great demand of the time. Product lines introduced two characteristics: platform and variability (POHL; BÖCKLE; LINDEN, 2005). These concepts were also adopted for software product line development.

In the SPL context, a platform is a technology or process on which other processes or even other technologies are built. It is usually what is called “core of the system”, which contains the common features that are shared by all systems of that family. Variability refers to the flexibility, in other words, the ability to create artifacts to be reused among different products of the same family (KANG et al., 1990).

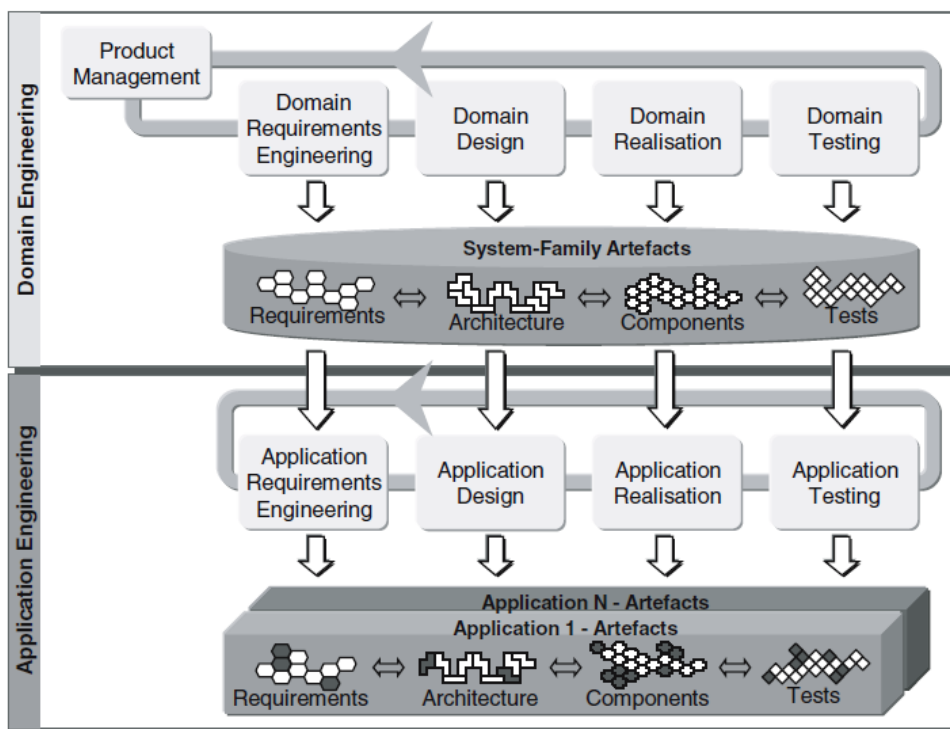
The SPL is defined by (CLEMENTS; NORTHROP, 2002) as “a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission”. Some of the main benefits of implementing SPL are the long-term cost reduction, significant quality improvement of the products, and reduction of the time-to-market (POHL; BÖCKLE; LINDEN, 2005). As it happens with regular software products, SPL also need a development process (KANG et al., 2005) (MARTINEZ et al., 2015) (OTSUKA et al., 2011) (STOERMER; O’BRIEN, 2001) (ZIADI et al., 2012). In this case, it is a special process because SPL cannot be developed as a regular software (LINDEN; SCHMID; ROMMES, 2007). The field dedicated to the study of this development is SPLE. However, a common practice in SPL development is to start from a set of product variants and extract their variabilities to create the SPL. This process is called Software Product Line Reengineering (ASSUNÇÃO et al., 2017). For both SPL reengineering and SPLE, a variability mechanism is used to manage the features of the SPL. This mechanism is usually a feature model, which can be created using different notations. SPLE, SPL reengineering, and feature model concepts are described in the following sections.

2.1.1 Software Product Line Engineering

SPLE is the paradigm responsible for the development and study of SPL. It utilizes platforms and mass customization concepts (DAVIS, 1989) to enable variability management. Plenty of technologies are used in SPLE as facilitators, making the SPL creation process easier. From these technologies, we can highlight the component-based architecture, software project patterns, and the object-oriented paradigm (LINDEN; SCHMID; ROMMES, 2007).

A model for SPLE, illustrated in Figure 1 was proposed by (POHL; BÖCKLE; LINDEN, 2005); this model is divided between domain engineering and application engineering, which are described below.

Figure 1 – The two-life-cycle model of SPLE.



Source: (POHL; BÖCKLE; LINDEN, 2005)

2.1.1.1 Domain Engineering

The life-cycle of Domain Engineering focuses on the definition of the SPL scope and common assets generation, from requirements to testing, that compose the SPL platform. We present a brief description adapted from (POHL; BÖCKLE; LINDEN, 2005) of each sub-process below:

- **Product Management:** the product management sub-process aims to define the scope of the SPL and manages the organization of the product portfolio. This

process generates the product roadmap, defining the common and variable features of the products. It also generates the schedule of each product release and the list of reusable artifacts;

- **Domain Requirements Engineering:** the product roadmap is used to elicit, document, negotiate, validate and verify, and manage the requirements, a set of common variables, well-defined requirements, and the SPL variability model;
- **Domain Design:** domain requirements are used alongside the SPL variability model to define the Software Product Line Architecture (SPLA);
- **Domain Realisation:** the SPLA and other domain assets are analyzed to design and implement the software components of the SPL domain;
- **Domain Testing:** it is when the realization components are tested alongside their specifications. The main goal is to create reusable test artifacts, aiming to reduce cost and effort during application testing.

2.1.1.2 Application Engineering

In the Application Engineering life-cycle, the common and variable assets developed in the Domain Engineering are combined with specific assets to create a SPL product. The application engineering is composed of four sub-processes are briefly described as follows:

- **Application Requirements Engineering:** the domain requirements and the SPL roadmap are used alongside the target application's specific requirements. This sub-process generates the requirements of a specific product.
- **Application Design:** the architecture of a specific application is created by using the SPLA. This is done by selecting the desired parts from the SPLA and adding some adaptations related to the specific product.
- **Application Realisation:** the main goal is to generate an executable software product. This generation is done by using the application architecture and the domain realization. At the end, a software product should be generated by simply selecting components to realize each interface.
- **Application Testing:** here, the software product generated is tested by utilizing the domain test artifacts and the application realization product.

2.1.2 Software Product Line Reengineering

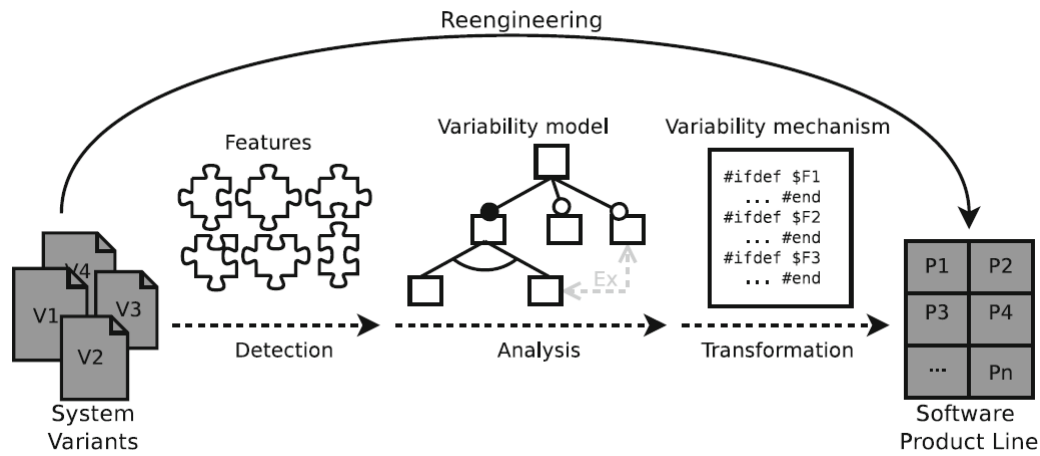
According to (KRUEGER, 2001), there are three possible approaches to develop SPL: proactive, reactive, and extractive. By using the proactive approach, an organization plans, analyzes, designs, and develops a complete SPL including the whole scope of their products. With the reactive approach, the organization incrementally increases its SPL development to reach the demand for new products or emerging new requirements. However, the extractive approach 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. An extractive approach is the most indicated when the organization already has a set of software variants because it is easy to identify and extract the commonalities and the variabilities among them. This process of extraction is called SPL 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; CROSS, 1990). In the SPL context, reengineering is used to transform a system, system family, or system variants into a SPL as illustrated in Figure 2. According to (ASSUNÇÃO et al., 2017), the SPL reengineering process is composed of three main phases, described below:

- Detection: it is when variabilities and commonalities, represented in the form of features, of the products are identified and extracted throughout the use of feature retrieval techniques. Techniques and methods used during this phase aim to extract data from artifacts, such as class diagrams and source code;
- Analysis: it is where the information extracted is used to design and organize the functional features into a variability model, usually a feature model;
- Transformation: it is when artifacts linked to these features, such as source code and requirements list, are managed, refactored, and modified to create the SPL.

In this work, we propose a generic framework to support this planning process. In this sense, we define planning the SPL reengineering as “*to extract, categorize and group features from legacy systems, as well as to design a Feature Model (FM)*”. Hence, we also refer to this planning phase as feature retrieval and analysis. Thus, our proposal gives support to the first two phases of the reengineering process defined by Assunção et al. (2017). In addition, we also consider the SPL scope definition to be part of SPL planning. During this process, feature retrieval techniques are used to retrieve the variabilities and commonalities of the products.

Figure 2 – SPL Reengineering process.



Source: (ASSUNÇÃO et al., 2017)

2.1.3 Feature Retrieval Techniques

Based on Assunção et al. (2017), we classified the feature retrieval strategies in three groups: i) static analysis: clustering, dependency analysis, and data-flow analysis; ii) information retrieval: formal concept analysis, latent semantic indexing, and vector space model; iii) support strategies: expert-driven extraction, heuristics, and rule-based extraction. Assunção et al. (2017) also identified the use of dynamic analysis strategies. However, these strategies are not considered for Prepare Assemble and Execute Framework for Software Product Line Reengineering (PAXSPL) due to their technological requirement, which would reduce the adaptability of our proposal.

The first strategy is Static Analysis, done by analyzing a program without its execution (CHRISTENSEN; MØLLER; SCHWARTZBACH, 2003). Information analyzed may include structural information and static artifacts. Static analysis techniques are recommended when analyzing source code. Furthermore, it is recommended that the used source code possesses low coupling and high cohesion. There are a lot of different Static Analysis techniques, from which we highlight:

- Clustering: groups a set of objects (*e.g.*, features) based on their similarities in clusters (JAIN; DUBES, 1988);
- Dependency analysis: leverages static dependencies among program elements. It may be used to validate and describe the interdependence among elements (KLATT; KROGMANN; SEIDL, 2014);
- Data-flow analysis: gather information about possible values calculated at different points of a software system. This information is used to determine which parts of

that program a particular value might propagate (RYSSEL; PLOENNIGS; KABITZSCH, 2011).

The Information Retrieval strategy collects and analyzes information in artifacts considering text structure, text similarity, etc (FRAKES; BAEZA-YATES, 1992). Information retrieval techniques commonly use documents written in natural language. They are also generally used in requirements artifacts; however, they can also be used in source code. To do that, both the source code and the requirements must have meaningful names. Information retrieval techniques are commonly used alongside Static Analysis techniques such as clustering. From the Information Retrieval techniques, we emphasize on:

- FCA: a mathematical method that provides a way to identify “meaningful groupings of objects that have common attributes” (STUMME, 2009);
- Latent Semantic Indexing (LSI): an indexing and retrieval method that uses a mathematical technique to identify patterns in the relationships between the terms and concepts contained in an unstructured collection of text (DUMAIS, 2004);
- Vector Space Model (VSM): an algebraic model for representing text documents in a way where the objects retrieved are modeled as elements of a vector space (SALTON; WONG; YANG, 1975).

For the last group, support strategies, we have the Expert-Driven Extraction, Heuristics, and Rule-based techniques. Expert-Driven Extraction is based on the knowledge and experience of experts such as domain engineers, software engineers, and stakeholders. To apply the Expert-Driven strategy, it is highly recommended that the team should have skills and knowledge involving the SPL reengineering process. Heuristics are proposed by many authors (RUBIN; CHECHIK, 2012) (BÉCAN et al., 2013). They are used alongside other techniques to improve retrieval results. The Rule-based techniques (MU; WANG; GUO, 2009) are similar to heuristics because they are also used alongside other techniques. Usually, these rules are created to guide and help whoever is performing the feature extraction.

2.1.4 Feature Model

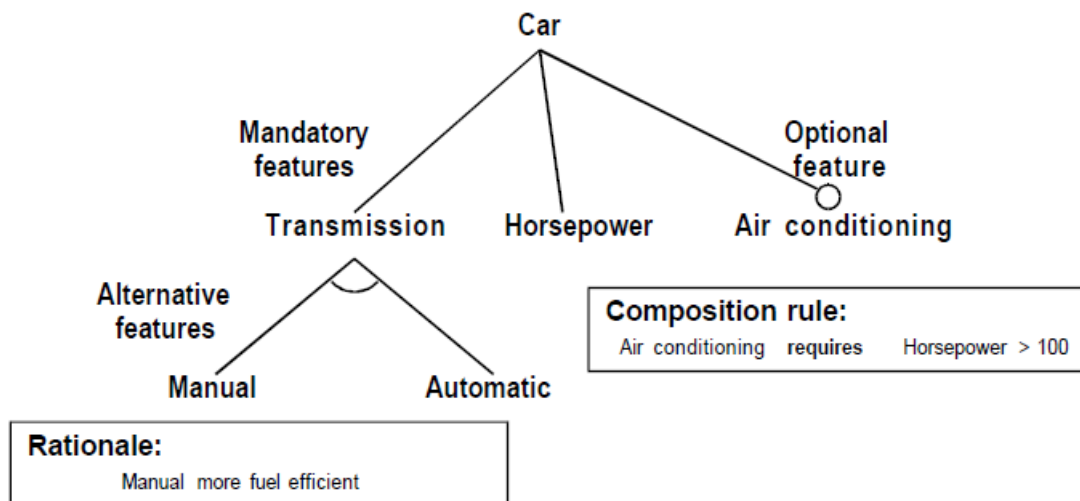
A feature model is one of the most used mechanisms to represent the SPL variability (CLEMENTS; NORTHROP, 2002). To the best of our knowledge the first feature model notation, Feature-Oriented Domain Analysis (FODA) was presented by (KANG et al., 1990). The feature model is represented in a tree structure, where its root, in the SPL context, is usually the SPL being modeled. This notation introduced some concepts that are shared by other feature model notation such as:

- Mandatory Features: features shared by all products of the SPL;

- Optional Features: features that can be present in the SPL realization or not based on the stakeholders choice;
- Alternative *xor-group*: these features are always part of a group where only one of them must be part of the product realization;
- Composition Rule: rules that define whether a feature will be *required* or *excluded* from the product;
- Rationale: these descriptive rules serve as indications of possible product realizations based on features selection.

A feature model constructed using the FODA notation is illustrated in Figure 3. The `car` represents the SPL, `transmission` and `horsepower` are mandatory features, `air conditioning` is an optional feature, and `manual` and `automatic` are an alternative *xor-group*. In addition, a composition rule and a rationale are shown.

Figure 3 – A feature model of a car.



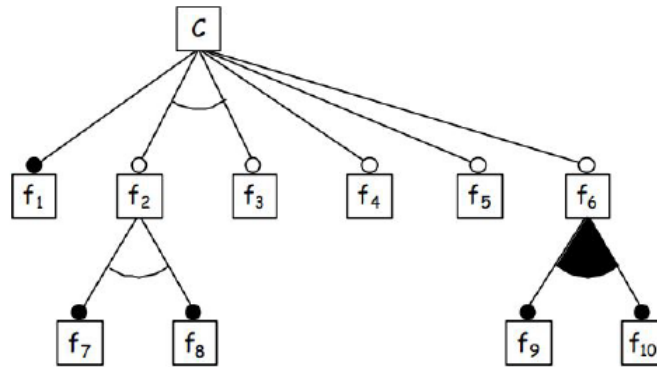
Source: (KANG et al., 1990)

Other notations, such as Generative Programming presented by (CZARNECKI; EISENECKER, 1999) added another important concept to feature models: *or-features*. The *or-features* are a group of features similar to alternative *xor-features*, however, in an *or-group*, more than one feature can be selected. An example of a Generative Programming feature model is presented in Figure 4, where f_9 and f_{10} represent an *or-group*.

2.1.5 SPL Scoping

Planning to apply SPL-based techniques considering business alignment and goals is called as Scoping (JOHN, 2010). Literature abounds with proposals for SPL-based production plans as surveyed in John e Eisenbarth (2009). These works are devoted

Figure 4 – A feature model created using the Generative Programming notation.



Source: (CZARNECKI; EISENECKER, 1999)

to characterizing particular artifacts and tasks for specific domains. Proposals such as PuLSE (BAYER; MUTHIG; WIDEN, 2000) and RiPLE (BALBINO; ALMEIDA; MEIRA, 2011) are examples of scoping approaches well-known in the literature. Additional information about SPL scoping is described in Chapter 4.

2.2 Software Process

The term software process is defined by Feiler e Humphrey (1993) as “a set of partially ordered steps intended to reach a goal[...] For software development, the goal is production or enhancement of software products[...]”. In addition, (PRESSMAN, 2005) defines process as “a framework for the tasks that are required to build high-quality software”. As the definition may change, a process, in the software context, is used to create artifacts related to software development.

Despite its definition, some aspects related to software process may be considered its main areas of investigation. As presented by (FUGGETTA; NITTO, 2014), these areas are: process modeling and support, including techniques and languages to give support for the design of software processes; process improvement, which concerns the constant investigation to find points for improvement within a process; metrics and empirical studies, which includes the creation and use of quality metrics and execution of experiments and case studies to evaluate a process; “real” processes, a process which had passed by the three aspects mentioned early, also called “concrete” processes, such as the Rational Unified Process (KRUCHTEN, 2004).

The software process methodology may have to follow different strategies. There is the traditional software development process, which gives a similar priority for both software production and software documentation (PRESSMAN, 2005). We can also highlight agile software processes, usually intending to be iterative and focusing on software delivery (MARTIN, 2002), such as Scrum (SCHWABER; BEEDLE, 2002).

2.3 Organizational Scenarios

As discussed in (BOSCH, 2006; BOSCH, 2009), several challenges arise when developing SPL in intra-organizational scenarios. However, as the scope of SPL expands, the development of SPL would eventually transcend the organizational boundary. In this sense, when performing the reengineering of legacy systems, companies may face challenges related to the understanding of the features being extracted and modeled (HUBAUX; HEYMANS; BENAVIDES, 2008). This issue highlights the importance of a well-defined process for conducting the feature retrieval and analysis in different scenarios. A major motivation for our work is to narrow this gap. In this sense, is important to define and explain what are the definitions of these different scenarios.

We may define a scenario as “a certain group of engineers using a specific set of retrieval techniques for extracting features from a determined set of artifacts”. Following this definition, a possible scenario would be: “a group of three members of an organization, an analyst, a developer and a project manager, with previous knowledge in FCA, applying it to retrieve features from a requirements list”. In this particular scenario, the members have previous knowledge in a retrieval technique and used it to retrieve features from an existing artifact. In this sense, we established a very similar scenario by “destroying” the requirements list and providing the users with only the source code from the legacy system. By only applying this “minor” change, we would have a scenario where the FCA technique would be much harder to be applied in source code than in requirements. Thus, maybe the clustering technique would be more reliable in this new scenario.

By understanding what a scenario is and how it may vary from project to project, and also from company to company, we may argue that the guidance of how to perform SPL reengineering is needed for formalizing such process, mitigating many of the challenges discussed in the literature.

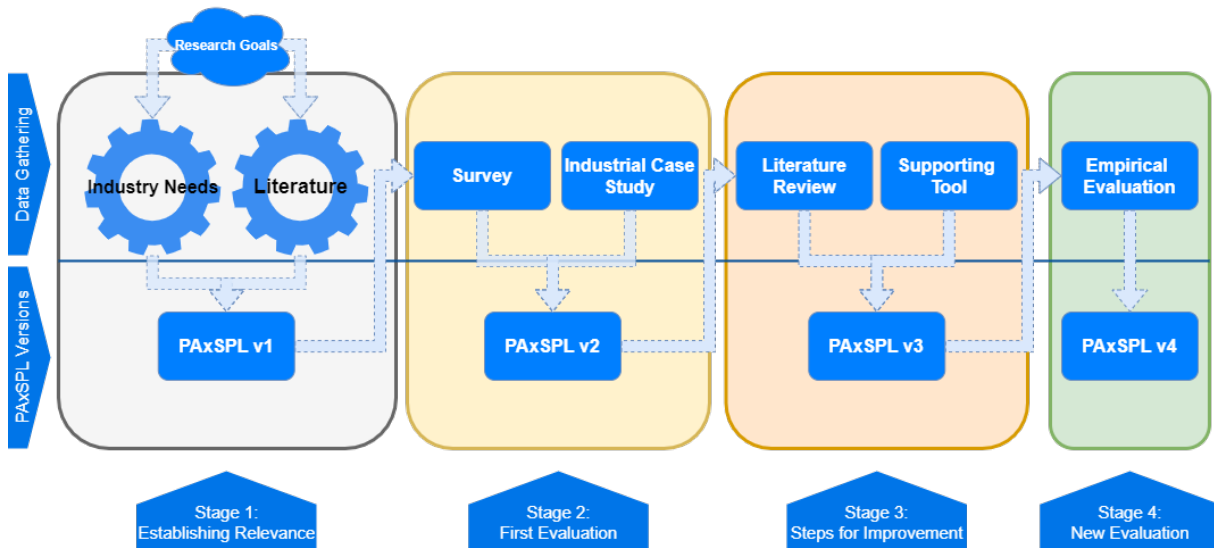
2.4 Chapter Lessons

In this chapter, we explored the main concepts of our work. The SPL, SPLE, SPL Scoping and SPL Reengineering process concepts are important for understanding our proposal. Nonetheless, the feature model and feature retrieval techniques and concepts are crucial for us to develop our process because both concepts must be acknowledged when we start the analysis of related work to extract such information. We noticed that are a different kind of techniques that can be used for different scenarios. The software process concept is important because we are designing our own process based on approaches found in the literature. Lastly, it is important to define organizational scenarios are these are part of the motivation for conducting this work.

3 METHODOLOGY

In this chapter, we describe the research methodology adopted to achieve our goals. This methodology was based on the evolutionary multi-method research approach presented by O’Leary e Richardson (2012). As illustrated by Figure 5, the research design of the methodology was divided into four different stages: Stage 1) Establishing Relevance; Stage 2) First Evaluation; Stage 3) Steps For Improvement; Stage 4) New Evaluation. We considered that each activity was executed for gathering data (The top portion of Figure 5). Then, a new version of PAXSPL process, now called PAXSPL, was generated for each stage. These stages and their activities are explained in the following sections.

Figure 5 – Research Methodology Design



Source: Author.

3.1 Establishing Relevance

Any research project needs to establish its relevance before being developed. As a strategy for collecting evidence about the relevance of our proposal, we first analyzed industry needs. Then, we looked into studies in the literature to understand how we could satisfy this need. We were able to obtain such answers from Assunção et al. (2017). By analyzing the data collected in their Systematic Mapping Study (SMS) we proposed the first version of PAXSPL. This version was then reported in Marchezan et al. (2017).

3.2 PAXSPL First Evaluation

The first evaluation, was based on a survey with students and practitioners of the field, and an industrial case study. The survey results were reported in Marchezan et al. (2017), while the case study was published in Marchezan et al. (2019b). Both evaluations were important for generating the second version of the process, but also for providing new aspects for improvement, which then generated PAXSPL v2.

3.3 Steps for Improvement

The aspects needing improvements identified in Marchezan et al. (2019b) are those addressed in this work. Thus, stages 3 and 4 of the methodology are the main focus of this work in particular, which will be expanding the research from prior years.

One of the main aspects identified in the survey was the need for having activities and guidelines aiming at addressing SPL scoping topics. Thus, we conducted a Systematic Literature Review (SLR) (KITCHENHAM et al., 2010) on SPL scoping, which is reported in Chapter 4.

Another main point for improvement addressed here is the need for a supporting tool. As discussed in Marchezan et al. (2019b) some activities and artifacts of PAXSPL v2 demanded considerable effort than others. Also, as all reports have to be filled manually (*e.g.* using Google Docs) the quality of those artifacts could be compromised. Due to these reasons, we decided to develop the PAXSPL supporting tool, presented in Chapter 5.

Lastly, PAXSPL v3, resulted from these improvements is presented in Chapter 5. However, as Figure 5 shows, we are currently at the last portion of Stage 3, thus, both the tool and PAXSPL v3 may still be modified after our new evaluation.

3.4 New Evaluation

The final stage of our research is an empirical evaluation. This evaluation was applied in PAXSPL v3, which already has its supporting tool. The main goal is to evaluate how PAXSPL is adaptable in adherence to different scenarios extracted from similar studies found in the literature. The evaluation design, procedure, results, and discussions are presented in Chapter 6. Based on the data collected from the evaluations, we performed improvements into the process, guidelines, and supporting tool generating PAXSPL v4.

4 A SYSTEMATIC LITERATURE REVIEW ON SPL SCOPING

In this chapter, we describe a SLR performed to gather the information to updated the state of the art in the SPL scoping field as well as to improve PAXSPL (MARCHEZAN et al., 2019b) in terms of SPL Scoping. Section 4.1 describes the protocol adopted for our SLR. and presents our findings. Section 4.2 discusses the approaches extracted from the literature to answer and discuss our Research Question (RQ)s.

4.1 SLR Design and Execution

An SLR protocol must be well planned and executed to obtain the desired results. We adopted a protocol proposed by Kitchenham et al. (2009). We used a SLR supporting tool (MARCHEZAN et al., 2019) for the execution of this review. The first step is to define its goal and RQs of our SLR. The main goal of our study is:

Goal: *Review the literature on SPL scoping for the purpose of identifying similarities and differences among existing approaches and processes, business aspects, conceptual characteristics, and research opportunities.*

To achieve this goal, we derive four Research Questions (RQs):

- RQ.1. **What are the similarities and differences among the approaches?** In this question we want to investigate the processes followed by each approach in terms of activities, steps, types of scoping, and how they are open for adaptation.
- RQ.2. **How are existing scoping approaches evaluated?** To reason about the applicability of scoping approaches, we investigate the empirical methods used in the studies and the domain where the evaluation had focus.
- RQ.3. **How is the decision making during the process of SPL scope definition?** Here we explore which are the factors that drive the decisions during the scoping definition. For example, which metrics and cost models are considered for choosing features to be prioritized, estimating the effort to implement variabilities, or maintaining many products in a business domain.
- RQ.4. **What are the open research gaps and opportunities for new studies on the topic of SPL scoping?** The objective of this research question is to shed light on new pieces of research on SPL scoping. We considered the findings of our SLR and the observed gaps in the primary sources to compose a list of directions for new studies.

For the search of primary sources, an important part of the SLR is the definition of the relevant terms, and their synonyms, related to the goal and RQs. To define this string,

we relied on the population, intervention, comparison, outcomes and context (PICOC) method (WOHLIN et al., 2012), as described next:

- **Population:** studies that deal with SPL scoping.
- **Intervention:** SPL scoping process, approaches, techniques, strategies, or similar proposals.
- **Comparison:** activities performed, scoping types covered, similarities and differences, cost models proposed or used, metrics for scoping, approach adaptation, evaluations applied, and domains.
- **Outcome:** a set of SPL scoping approaches, an overview of the approaches describing their activities, types of scoping, similarities and differences among them, cost models and metrics used for scoping, types of adaptation, evaluation methods, the domain of the systems used for evaluation, gaps and opportunities in the field.
- **Context:** both academic and industrial context. Considering the industrial context, we aim at identifying in which domains the approaches were applied.

Based on the PICOC information, we extracted terms and synonyms to construct the base search string, using *OR* and *AND* operators, as presented in Table 1. Table 2 presents the six digital libraries (DLs) used to search for primary sources and the specific search string for each DL. These specific search strings were defined after executing, analyzing, and refining the base search string until they returned a satisfactory result. Finally, for the screening of the studies returned from each DL, according to the protocol of our SLR, we defined one inclusion criterion (IC) and four exclusion criteria (EC) to select only relevant papers, as presented in Table 3.

Table 1 – Terms, Synonyms and the Search String

Terms	Synonyms
Scope	scoping
Software product line	product line, software family, software product family, software reuse, SPL
Approach	method, methodology, process, technique
Search String	Scope OR Scoping AND Software product line OR product line OR software family OR software product family OR software reuse OR SPL AND approach OR method OR methodology OR process OR technique

Source: Author.

The next step in the SLR protocol is to evaluate the quality of the selected studies. In this way, we defined a set of quality assessment questions (QAs) that were used to qualify and classify the studies. We assigned a score for each question, and after

Table 2 – Digital Libraries and Search Strings

Digital Library	Search String
ACM	(Scope Scoping) AND (“Software product line” “product line” “software family” “software product family” “software reuse” SPL) AND (approach method methodology process technique)
Eng. Village	((Scope OR Scoping) WN KY) AND ((“Software product line” OR “product line” OR “software family” OR “software product family” OR “software reuse” OR SPL) WN KY) AND ((approach OR method OR methodology OR process OR technique) WN KY)
IEEE Xplore	(Scope OR Scoping) AND (“Software product line” OR “product line” OR “software family” OR “software product family” OR “software reuse” OR SPL) AND (approach OR method OR methodology OR process OR technique)
Science Direct	(Scope OR Scoping) AND (“Software product line” OR “product line” OR “software family” OR “software product family” OR “software reuse” OR SPL) AND (approach OR method OR methodology OR process OR technique)
Scopus	TITLE-ABS-KEY (Scope OR Scoping) AND (“Software product line” OR “product line” OR “software family” OR “software product family” OR “software reuse” OR SPL) AND (approach OR method OR methodology OR process OR technique)
Web of Science	TS=((Scope OR Scoping) AND (“Software product line” OR “product line” OR “software family” OR “software product family” OR “software reuse” OR SPL) AND (approach OR method OR methodology OR process OR technique))

Source: Author

Table 3 – Inclusion/Exclusion Criteria

Inclusion Criteria	Exclusion Criteria
IC1. The primary study must present a SPL scoping approach or similar proposal.	EC1. Studies written in languages other than English.
	EC2. Studies not available online.
	EC3. Studies without any evaluation of the approach.
	EC4. Studies with less than 6 pages.

Source: Author

answering them, we computed the final score for each study. The possible answers for each question are: “*total*”, when the study has information to answer the QA, then the question scores 1 point; “*partial*”, when we can use the study only to partially answer the QA, the score, in this case, is 0.5; and “*none*” when the study does not provide any information to answer the question, having a score equal to 0. The QAs and their respective score rules are described next.

Quality Assessment Questions:

QA1. Does the study describe the activities and artifacts regarding SPL scoping?

Total: the study details its activities and artifacts presenting examples of use;

Partial: the study only presents a list of brief description of its activities and artifacts;

None: neither activities nor artifacts are presented.

QA2. Does the study present some kind of process/work-flow?

Total: the study presents a well-documented process containing roles, artifacts and a work-flow among its activities;

Partial: a work-flow for performing the activities is presented;

None: no work-flow is defined.

QA3. Does the study present an evaluation and its results?

Total: the study applied and reported an empirical evaluation discussing its results;

Partial: the study applied and reported a non-empirical evaluation discussing its results;

None: no evaluation is applied or discussed.

Another important artifact of an SLR is the data extraction form. The content retrieved when applying this extraction into the studies is crucial for achieving our goals. Our data extraction (DE) form had the following fields:

DE1. Title;

DE2. Author;

DE3. Year of publication;

DE4. Study goals;

DE5. Study summary;

DE6. Scoping Activities;

DE7. Scope types;

DE8. Scoping metrics;

DE9. Cost model strategies;

DE10. Adaptation strategy;

DE11. System domains and cross domains;

DE12. Proposal evaluation type;

DE13. Study limitations;

DE14. Research opportunities identified/proposed by the authors.

After performing the search in the digital libraries, a total of 851 studies were retrieved. Table 4 presents the number of studies retrieved from each DL. From these studies, 152 were duplicated. During the application of the I/E criteria, 36 studies remained. These studies were used as input for the snowballing (WOHLIN, 2014) which resulted in the inclusion of 9 studies. Thus, the final set of studies analyzed to answer our questions was composed of 45 primary sources¹.

Table 4 – Numbers of the SLR on Scoping

Source	#
ACM	67
Engineering Village	161
IEEEExplore	191
Science Direct	14
Scopus	350
Web of Science	68
Retrieved Studies	851
Duplicates Removed	152
I/E Criteria	36
Added by Snowballing	9
Selected Studies	45

Source: Author.

4.2 Results and RQ Answers

In this section, we present and discuss the results obtained after the reading and data collection of the primary sources. These results are the basis to answer the four posed RQs.

4.2.1 RQ1. What are the similarities and differences among the approaches?

Among the 45 primary sources, we found 33 approaches. Once that some approaches are presented by more than one article, it is important to highlight the main contribution of each one. Thus, the 45 primary studies are summarized in Table 5, alongside their goals.

¹ The repository of the SLR is available at <https://github.com/lucianoMarchezan/lesseResearch/tree/master/slr_SPL_scoping>

Table 5: Studies and Goals

Study	Proposal Name	Study Goal
Bayer, Muthig e Widen (2000)	PuLSE-CDA	Presents CDA, a method for domain analysis that is customizable to the project context where it will be applied.
deBaud e Schmid (1999)	PULSE-Eco	PULSE-Eco, a technique especially developed to address product line scoping.
Bayer et al. (2000)	PuLSE-I	Presents the application engineering process associated with the PuLSE SPLE method.
Knauber et al. (2000)	PuLSE	Applies PuLSE method in six small and medium-sized enterprises addressing six different domains
Kishi, Noda e Katayama (2002)	Kishi <i>et al.</i>	Proposes a method for product line scoping as a decision-making activity in which we determine the appropriate scope of the product line considering both the whole and the individual optimalities.
Schmid (2000)	PuLSE	Presents a framework that helps to assess how information elicited during scoping is done at the beginning of product line scoping.
Schmid (2002)	PuLSE	Presents improvements performed in PuLSE-Eco.
Schmid et al. (2005)	PuLSE	Discusses experiences with a project where researchers successfully dealt with SPL scoping difficulties and achieved a successful product line transition.
Park e Kim (2005)	Park <i>et al.</i>	Proposes a process for domain analysis and economical analysis of core asset scope. They also defined guidelines for each activity of the process.
Ramachandran e Allen (2005)	FARE	Introduces a method for analyzing requirements for their scope and for their potential to be candidate requirements for a product family.
John et al. (2006)	PuLSE	Authors present an update of the PuLSE process to include more detailed context characteristics.
Her et al. (2007)	Her <i>et al.</i>	Proposes a comprehensive framework for evaluating the reusability of core assets in SPL based on ISO/IEC 9126.
Noor, Grünbacher e Briggs (2007)	Noor <i>et al.</i>	Proposes a collaborative product line scoping approach for reengineering-based product line adoption which is based on involving success-critical stakeholders to balance business and technical concerns.
Noor, Grünbacher e Hoyer (2008)	Noor <i>et al.</i>	Describes a collaborative scoping approach for organizations migrating existing products to a product line
Kim, Park e Sugumaran (2008)	DRAMA	Provides a framework for modeling domain architecture based on domain requirements within SPL.
Carbon et al. (2008)	Planning Game for SPLE	Presents an adaptation of the agile practice “planning game” to a real SPL context.

Table 5: Continued

Study	Approach Name	Study Goal
Estublier, Dieng e Leveque (2010)	CADSE	Describes how the evolution of the associated engineering environment market and SPL scope needs are addressed together.
John (2010)	CAVE	Describes the CAVE approach and its industrial applications as a solution for the domain expert lack of involvement problem.
Ullah, Ruhe e Garousi (2010)	COPE+	Introduces a method that attempts to address shortcomings for the specific evolution scenario when a single evolving software system is evolved into SPL.
Elsner et al. (2010)	PLiCs	Provides a generic, reusable reference architecture and methodology for implementing such customizable product lines.
Villela, Dörr e John (2010)	PLEvo-Scoping	Describes a quasi-experiment performed to characterize PLEvo-Scoping in terms of adequacy and feasibility
Mærsk-Møller e Jørgensen (2010)	PuLSE-Eco	Reports experience from applying SPLE in a small team.
Cavalcanti et al. (2011)	Cavalcanti <i>et al.</i>	Presents a metamodel that aims to coordinate SPL activities by managing different SPL phases and their responsibilities and to maintain the traceability and variability among different artifacts.
Balbino, Almeida e Meira (2011)	RiPLE SC	Proposes RiPLE-SC, an agile scoping process for SPL.
Muller (2011)	VB Portfolio Opt.	Introduces Value-Based Portfolio Optimization as an addition to common Product Portfolio Scoping approaches that help to decide on what features are most important to realize.
Acher et al. (2012)	Acher <i>et al.</i>	Aims at easing the transition from product descriptions expressed in a tabular format to FMs accurately representing them.
Bartholdt e Becker (2012)	Bartholdt <i>et al.</i>	Discusses experiences related with the scope extension of a SPL.
Gillain et al. (2012)	Gillain <i>et al.</i>	Proposes a mathematical program able to optimize the product portfolio scope of a software product line and sketch both a development and a release planning
O’Leary, Almeida e Richardson (2012)	Pro-PD	Defines a systematic process that provides a structured approach to the derivation of products from an SPL based on a set of tasks roles and artifacts.
Lobato et al. (2012)	RiPLE-SC	Aims at identifying SPL risks during the scoping and requirement disciplines to provide information to better understand risk management in SPL.

Table 5: Continued

Study	Approach Name	Study Goal
Abbas e Andersson (2013)	ASPLE	Proposes extensions to an architectural reasoning framework with constructs/artifacts to define and model a domain's scope and dynamic variability
Cruz et al. (2013)	Cruz <i>et al.</i>	Presents a hybrid approach which combines fuzzy inference systems and the multi-objective metaheuristics
Souza et al. (2013)	SPLSmart	Aims at gathering evidence about the effects of applying an inspection approach to feature specification for SPL.
Nöbauer, Seyff e Groher (2014)	Nobauer <i>et al.</i>	Presents an evaluation of a tool-supported approach that enables the semi-automatic analysis of existing products to calculate their similarity.
Silva et al. (2014)	RiPLE-SC	The study is a step towards bridging the gap of SPL for the situation for small to medium-sized enterprises in contextual evidence by characterizing the weaknesses discovered in the scoping (SC) and requirements (RE) disciplines of SPL.
Sierszecki et al. (2014)	Sierszecki <i>et al.</i>	Presents an extension of the variability management that goes beyond the scope of software assets reuse previously introduced into the organization.
Khtira, Benlarabi e Asri (2014)	SPLBench	Proposes a requirement-based framework that capitalizes on the specific products already derived from the product line.
Alsawalqah, Kang e Lee (2014)	PPSMS	Proposes a novel method to find the optimized scope of a software product platform based on end-user features
Ianzen et al. (2015)	Ianzen <i>et al.</i>	Presents a semi-automatic approach for defining scope identification and classification of product features along with an approach for evaluating the variabilities and commonalities between the established line and a new product.
Karimpour e Ruhe (2016)	Karimpour <i>et al.</i>	Proposes to include uncertainty as part of the SPL scoping model. Scoping planning developed in consideration of uncertainty would be more robust against possible fluctuations in estimates.
Neto et al. (2016)	Neto <i>et al.</i>	Presents an improved hybrid approach to solve the feature model selection problem aiming at supporting product portfolio scoping.
Alam, Khan e Zafar (2017b)	ISPL	Proposes an improved framework for SPL which addresses cross-cutting concerns such as security and configurability.
Alam, Khan e Zafar (2017a)	ISPL	Presents a validation of ISPL, using the Expert Opinion Technique.
Ojeda et al. (2018)	CoMeS	Proposes a collaborative method for SPL Scoping.

Table 5: Continued

Study	Approach Name	Study Goal
Ojeda, Rodriguez e Collazos (2019)	Small-SPL	Reports an exploratory study aimed to identify problems related to the collaborative work at scoping SPL in practice.

Source: Author

With the intent to derive a conceptual map and also a generic process, in the following, we analyzed each of these 33 approaches. They were grouped chronologically to better understand their evolution to accomplish specific scoping issues.

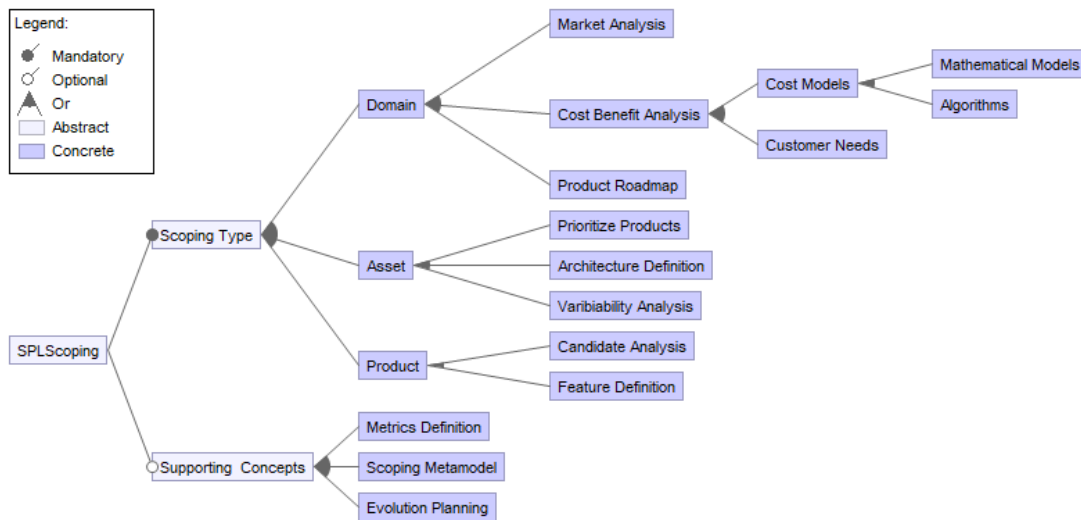
4.2.1.1 Approach Similarities and Differences

To investigate their characteristics, we analyzed similarities and differences between the 33 found approaches. They are composed of different characteristics that may or may not be necessary in a given context, for example, the use of cost models. Thus, we formulated a SPL scoping concept map. The research and practice in the area would benefit from a model that, both provide means for decision support and that constraint selection by a well-characterized set of features found in these 33 approaches. The Feature Model is a well-accepted representation for such a need (SEIDL; WINKELMANN; SCHAEFER, 2016), as illustrated by Figure 6, which describes contexts where each approach can be inserted, in a conceptual map. We derived this conceptual map to guide software engineers in the selection process for an approach, as well as to characterize the studies.

Our conceptual map is composed of two main concepts, *scoping type* that might be the domain, asset or product, and *supporting concepts*. The supporting concepts, metrics, metamodel, and evolution planning may be used to provide additional information and depth to any concept within the scoping type. For each scoping type, we have or-alternative concepts, such as feature definition and candidate analysis for Product Scoping; Variability analysis, architecture definition and prioritize products for Asset scoping; and cost-benefit analysis and market analysis for domain scoping. Cost benefits have customer needs and cost models, which may be mathematical models or algorithms.

Table 6 presents the traceability between each concept from the concept map (depicted in Figure 6) to the 33 approaches identified in this SLR. When considering the support concepts, 22 of them make use of such concepts, most are metrics definition (13), and evolution planning (11). Scoping meta-modeling is only considered in five approaches. Considering similarities in this aspect, both metric definition and evolution planning are present in six approaches (SPLSmart, COPE+, PLEvo-Scoping, RIPLE-SC, PPMS, and Karimpour *et al.*). However, the way they handle these concepts may vary, as discussed

Figure 6 – SPL Scoping Concept Map



Source: Author.

in Section 4.2.1.2.

In terms of scoping specific concepts, the product roadmap is present in ten approaches, while market analysis is present in the great majority (16). An important contrast between these two concepts is also presented in Table 6 as most of the approaches using product roadmap also make use of market analysis. This relation is due to roadmap containing market-related aspects, such as time for launching the software or related costs. A similar analysis may be performed considering the customer needs, as it's a concept that may also be related to both the product roadmap and market. Cost models, on the other hand, presented fewer relations with the other scoping domain concepts in this sense.

Considering the asset scoping concepts, we noticed that most approaches (20) apply variability analysis as part of SPL scoping. Architecture definition is also considered in several approaches (14). For approaches that apply both of these concepts, they are strongly related, as the variability is defined based on the reference architecture. This relation is also present in approaches that consider both variability analysis and prioritize products. Prioritize products is also related to the candidate analysis, a product scoping concept. It was found five works using the analysis of candidates for prioritizing products, and despite their different strategies, both concepts influence each other. A similar influence was found between the feature definition and the variability analysis concepts, as usually when defining how the variability of the asset is managed, approaches also look into how the features were defined for the SPL.

Although the majority of approaches (18) handle all types of scoping, their strategies vary. It is also important to mention that some approaches focus on specific contributions besides discussing scoping activities, such as CADSE focusing on a SPL scoping

Table 6 – Traceability of Scoping Concepts in the Primary Studies

Ref.	MD	SM	EP	PR	MA	CM	CN	PP	AD	VA	CA	FD
PuLSE			✓	✓	✓		✓		✓	✓	✓	✓
Kishi <i>et al.</i>								✓	✓		✓	
SPLSmart	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓
Park <i>et al.</i>	✓					✓						
FARE					✓	✓						
Her <i>et al.</i>	✓								✓	✓		✓
Noor <i>et al.</i>	✓			✓			✓	✓		✓		
DRAMA					✓	✓	✓	✓	✓			
Planning Game in SPLE			✓				✓	✓				
CADSE		✓	✓							✓		✓
COPE+	✓		✓								✓	✓
PLEvo-Scoping	✓		✓	✓	✓					✓		✓
CAVE				✓	✓					✓		✓
PLiCs		✓					✓		✓	✓		✓
Cavalcanti <i>et al.</i>	✓	✓							✓	✓		✓
RiPLE-SC	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓
VB Portfolio Opt.		✓			✓	✓	✓				✓	✓
Acher <i>et al.</i>		✓								✓	✓	✓
Bartholdt <i>et al.</i>					✓		✓		✓	✓	✓	✓
Gillain <i>et al.</i>					✓	✓	✓					✓
Pro-PD									✓	✓		✓
ASPLe				✓					✓	✓		
Cruz <i>et al.</i>	✓					✓	✓	✓		✓	✓	✓
Nobauer <i>et al.</i>				✓	✓		✓					
Sierszecki <i>et al.</i>			✓						✓	✓		✓
SPLBench	✓						✓		✓	✓		✓
PPSMS	✓		✓		✓	✓	✓	✓		✓	✓	✓
Ianzen <i>et al.</i>										✓	✓	✓
Karimpour <i>et al.</i>	✓		✓		✓	✓	✓					✓
Neto <i>et al.</i>	✓					✓					✓	✓
ISPL			✓	✓	✓		✓		✓	✓		✓
CoMeS				✓	✓		✓	✓				✓
Small-SPL					✓		✓					✓

MD - Metrics Definition; SM - Scoping Meta model; EP - Evolution Planning; PR - Product Roadmap; MA - Market Analysis; CM - Cost Modes; CN - Customer Needs; PP - Prioritize Products; AD - Architecture Definition; VA - Variability Analysis; CA - Candidates Analysis; FD - Feature Definition.

Source: Author

meta-model. More details about these aspects are discussed in the following sections.

4.2.1.2 Approach Activities

The concept map shown in Figure 6 is created based on the analysis of the activities performed by each approach, which were briefly described in Table 5. Next, the approaches are presented in more detail, allowing us to identify similarities and differences, and reasoning about a common process. To highlight the evolution of SPL scoping contributions, we present the approaches in chronological order. The year of the publications are presented in Table 7:

PuLSE (BAYER; MUTHIG; WIDEN, 2000; DEBAUD; SCHMID, 1999; BAYER et al., 2000; KNAUBER et al., 2000; SCHMID, 2000; SCHMID, 2002; SCHMID et al., 2005; JOHN et al., 2006; MÆRSK-MØLLER; JØRGENSEN, 2010): developed as a customizable method to support the conception, construction, usage, and evolution of SPL, PuLSE is divided in components. One of such components was

created specifically for SPL scoping, the PuLSE-Eco (DEBAUD; SCHMID, 1999). This component aims at identifying the characteristics that are directly supported by the reference 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 them to provide rough guidance on implementation aspects of the project.

Kishi *et al.* (KISHI; NODA; KATAYAMA, 2002): two types of requirements are considered for scoping: those for a unique product and those for the product line. The requirements for a unique product consist of functionality and quality attributes. The activities may be described as: i) identify the requirements for each product and the product line; ii) define the design policy in terms of priority among the requirements; iii) list the architectural candidates for the given products; iv) determine the preference of each architectural candidate and examine their applicability for each product; v) examine the candidates for the SPL scope, determining the preference of the candidates.

Park *et al.* (PARK; KIM, 2005): a process accompanied by guidelines for domain analysis and economic analysis of core assets scope. To analyze the economic value of the scope, the authors considered not only the variability but also dependencies among variabilities. Activities of the process: i) commonality analysis; ii) variability analysis; iii) variability dependency analysis; iv) domain model refinement; and v) economical evaluation of core asset scope.

FARE (RAMACHANDRAN; ALLEN, 2005): a method for analyzing requirements for their scope, and their potential to be candidate requirements for a SPL. To adapt the scope, commonality, and variability analysis to capture, organize, and managing requirements, the FARE method provides a systematic process, clearly specified checklists for self-assessment and improvement, cost-benefit analysis. It also provides method-specific guidelines for the use case method. The process is divided into: i) prepare; ii) plan; iii) commonality and variability analysis; iv) quantify; and v) review.

Her *et al.* (HER *et al.*, 2007): the proposed framework is based on ISO/IEC 9126 quality standard, and it consists of quality attributes, metrics for them, and guidelines for applying the metrics. The framework does not describe specific activities, however.

Noor *et al.* (NOOR; GRÜNBACHER; BRIGGS, 2007; NOOR; GRÜNBACHER; HOYER, 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 inappropriate 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 built upon it; v) prioritize product map: voting is performed to ascertain the priority of products and features of the product map, reasons for disagreements are revealed, and the consensus is built.

DRAMA (KIM; PARK; SUGUMARAN, 2008): DRAMA is a framework designed to aid the modeling of domain architecture. The framework consists of processes, methods, and a supporting tool for domain requirements analysis and domain architecture modeling. Covered scoping activities are divided into phases: i) identifying components: define business strategy and marketing plan; ii) calculating the priority of components: generates goal tree and components; iii) calculating the priority of quality attributes: prioritize components; iv) modeling domain architectures: analyze priority of quality attributes and define domain architectures.

Planning Game in SPLE (CARBON et al., 2008): a well know planning game used in traditional software development is adapted to a SPL context. The planning game is used as a means to provide to family engineers with feedback from the application. The game is subdivided into three phases with several activities: i) exploration phase: the application engineers write down their feedback on the available reusable artifacts employing reuse stories and prioritize them; ii) commitment phase: the scope of the next release of the reusable product line components is derived; iii) steering phase: After the scope of the next release has been fixed iterations to realize the selected reuse stories can be conducted.

CADSE (ESTUBLIER; DIENG; LEVEQUE, 2010): proposes a solution that uses meta-modeling and engineering environment evolution generation, scope composition, and market evolution. In addition, presents a component database and a selection language for the product variability evolution. The proposal does not present any flow of activities. CADSE, however, allows us to define and compose wide scope environments. The composition technology provides flexibility allowing to expand the scope of a given family with new concepts, new components, new features, and new constraints.

CAVE (JOHN, 2010): offers a solution concerning the problem of domain experts' availability during the SPLE. The approach is divided into three phases: i) preparation, where the SPL consultants collect user documentation; ii) analysis, when the SPL consultant applies patterns searching the documentation; iii) validation, domain experts validate and change the invalidated artifacts.

COPE+ (ULLAH; RUHE; GAROUSI, 2010): a decision support method to address the product evolution problem. From a high-level standpoint, the decision support method COPE+ has three modules. To analyze and update the data and results, it involves human decision making at various stages of the decision support. Scoping is

covered by the following activities: i) voice of the customer analysis: customer voting and prioritizing features; ii) structural Impact Analysis: identification of features impact in evolving software systems; iii) similarity Analysis: conformance of product variant implementations;

PLiCs (ELSNER et al., 2010): provides a tool-supported methodology for combination of SPL components based on customer needs. The tool follows a four steps process: i) specify customized SPL (CPL); ii) setup the CPL; iii) specify the CPL products; iv) generate the CPL product.

PLEvo-Scping (VILLELA; DÖRR; JOHN, 2010): complements and extends SPL scoping approaches by helping the SPL scoping team to anticipate emergent features and distinguish unstable from stable SPL features. It contains several activities: i) preparation for volatility analysis: establishing 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 time-frame, 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.

Cavalcanti et al. (CAVALCANTI et al., 2011): a metamodel providing support for several SPL aspects, such as scoping, requirements, tests, and project and risk management. There is no process itself, but the proposed metamodel may be used for various Scoping tasks: feature modeling, product scoping, variability management, and asset definition.

RiPLE-SC (BALBINO; ALMEIDA; MEIRA, 2011; LOBATO et al., 2012; SILVA et al., 2014): the main focus is to operate in small to medium-sized companies. The general process is divided into 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 prioritize the features on the product map.

Value-Based Portfolio Optimization (MULLER, 2011): the goal is to aid the planning of an optimal product portfolio, pricing the SPL products, and understanding the contribution of assets to profit. The authors provide an ontology and an algorithm to support decision making concerning product portfolio decisions. The approach may be SPLit into different steps: i) determine a profit optimal product portfolio; and ii) identify which assets have the highest positive impact on profit due to the products related to them.

Acher et al. (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 supporting scoping activities, which is used to parameterize the features extraction. The scoping is only covered as a single task. A practitioner may scope the data in various ways and for many purposes using the proposed language.

Bartholdt *et al.* (BARTHOLDT; BECKER, 2012): in the work, the authors discuss their experiences on the various activities of SPLE while extending a SPL. The authors describe how they have identified beneficial sub-domains to increase the commonality of their SPL while extending the scope of the SPL. Although the activities are not described as a workflow, the experience reports several activities performed such as commonality/variability analysis, use of supporting tools for mapping the domain, selection, and prioritization of features, and definition of a reference architecture.

Gillain *et al.* (GILLAIN *et al.*, 2012): a mathematical model based on the joint use of goals and features. The approach may help to optimize SPL scoping, especially product portfolio and assets scoping. During scoping, identification and evaluation of the three types of scoping can be performed separately. In general, is composed of three activities: i) determine relevant customers and what their needs are; ii) defining what the products are constituted of; iii) identify conditions for the product to realize the tasks.

Pro-PD (O'LEARY; ALMEIDA; RICHARDSON, 2012): a systematic process that provides a structured approach for the derivation of products from a SPL, based on a set of tasks, roles, and artifacts. The process is composed of several activities: i) initiate project; ii) identify and refine requirements; iii) derive the products; iv) develop the product; v) test the product; and vi) management and assessment.

ASPLe (ABBAS; ANDERSSON, 2013): the authors extended the architectural reasoning framework (ARF) to provide models and constructs to domain architects. These models allow reasoning 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.

Cruz *et al.* (CRUZ *et al.*, 2013): a systematic approach to deal with SPL optimization, based on several measures and the relevance perceived by the customers. Fuzzy inference systems and multi-objective optimization are used together to select the best products of different segments of users and group them in a portfolio. This goal is achieved by following five activities: i) inferring the cost of each asset; ii) calculating the asset relevance for each segment; iii) calculating candidate products for each segment; iv) qualifying candidate products; v) grouping the best product of each segment.

SPLSmart (SOUZA *et al.*, 2013): presents findings of how the inspection should be handled in a SPL scenario, to gather evidence of inspection on scoping artifacts,

as well as, identify possible gaps that have not been addressed by current research. The study was conducted based on an empirical study, aimed at investigating inspection-related issues during the application of a systematic software inspection approach in a SPL project. The inspection steps are: i) planning; ii) preparation; iii) meeting; iv) correction; and v) validation.

Nobauer *et al.* (NÖBAUER; SEYFF; GROHER, 2014): presents an evaluation of a tool-supported approach that enables the semi-automatic analysis of existing products to calculate their similarity. This is achieved by identifying key information on the reuse potential of existing software product configurations. The approach consists of the following steps: i) select products for analysis; ii) define the scope of the analysis; iii) define how similarity between selected configuration settings are calculated; iv) perform similarity analysis; v) draw conclusions.

Sierszecki *et al.* (SIERSZECKI *et al.*, 2014): concepts for further evolution of a SPL approach from code-centric to a more holistic approach are discussed. Authors show that moving further up by extending the variability management scope to requirements and portfolio management is a challenging task. They also outline a solution to this challenge and present a prototype realization. This realization is composed of: i) customer requirements; ii) product requirements; iii) software packages; iv) Implementation; v) variant validation; vi) qualification test; and vii) software release.

SPLBench (KHTIRA; BENLARABI; ASRI, 2014): the framework uses early customer requirements capitalizing on the specific products already generated. This information is used for providing metrics that aid stakeholders to make decisions. The framework is composed of several tasks: i) requirements stage, where elicitation, weighting, and transformation of requirements language are performed; ii) benchmarking; iii) features Stage, when the transformation of domain feature model do XML, and instantiation of application feature models are executed.

PPSMS (ALSAWALQAH; KANG; LEE, 2014): mathematically formulates optimized product platform scope that will maximize life cycle cost savings and the amount of commonality, while meeting the goals and needs of the envisioned customers segments. The method is divided into three main phases: i) analyzing customer needs using the Kano model and prioritize features; ii) analyzing features for potential commonality and variability; iii) optimization phase where a mathematical model is constructed, optimized with simulated annealing, and non-dominated solutions are analyzed.

Ianzen *et al.* (IANZEN *et al.*, 2015): a semi-automatic approach to assist in SPL scoping. The approach can help organizations that wish to migrate to the SPL approach to begin mapping their products and view them as families in the same domain that share components seen as common features. The activities may be described as: i) Scoping using a proposed method for the semi-automatic identification and classification of features based on artifacts of the organization's product; ii) realize product engineering,

facilitating the evaluation of the variabilities and commonalities between the created SPL and a new product.

Karimpour *et al.* (KARIMPOUR; RUHE, 2016): incorporates uncertainty in scoping optimization and its application to generate robust solutions. Captures uncertainty as part of the formulation and models scoping optimization as a multi-objective problem with profit and stability as fitness functions. The proposal is divided into different approaches: i) plan the portfolio scoping based on high profits goals using robust analysis algorithms; ii) incorporated uncertainty into SPL scope modeling; iii) perform optimization by simulating changes in the environment.

Neto *et al.* (NETO et al., 2016): a hybrid approach not dependent on any particular algorithm/technology. The approach can be split into several activities: i) calculate feature costs; ii) calculate feature relevance and generate candidate products; iii) calculate product suitability and select the best products.

ISPL (ALAM; KHAN; ZAFAR, 2017b; ALAM; KHAN; ZAFAR, 2017a): the model is a mix of an aspect-oriented and a feature-oriented approach. The first addresses crosscutting concerns and functional behaviors of SPL while the latter is used to capture variability and commonality of the products. The framework is composed of two high-level processes: domain engineering and application engineering. SPL Scoping is covered inside the domain engineering phase by the following activities: i) Potential SPL and its products are identified. ii) The scope should be practically attainable; iii) The output is a product portfolio, comprising of all potential products of the product family, and a product roadmap.

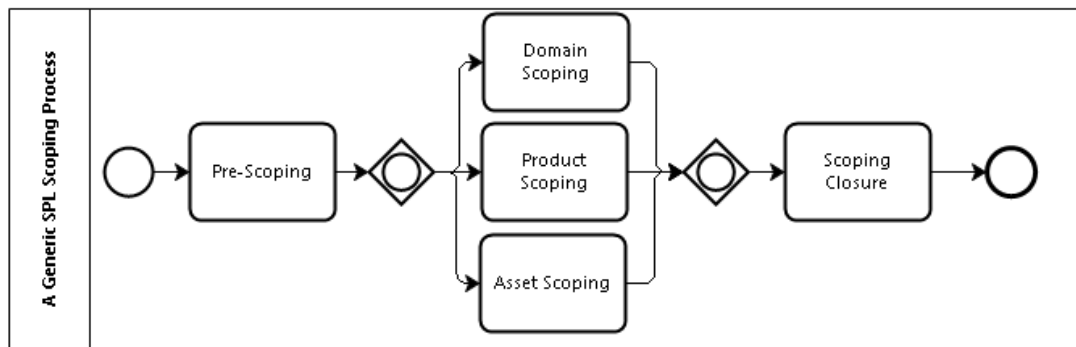
CoMeS (OJEDA et al., 2018): a collaborative way of guiding the definition of the scope, presents the tasks, and details the steps with the objective that the stakeholders' team could perform the outcomes of each task and at the end of the scoping activity, obtain a represented scope. The activities present 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.

Small-SPL (OJEDA; RODRIGUEZ; COLLAZOS, 2019): Small-SPL is a process for SPL engineering for small development companies based on the SEI's framework. The life cycle SPL includes three sub-processes: Domain Engineering and Product Engineering, geared by a third sub-process called Asset Management which is based on Requirements. The scoping activities covered are: i) Study the objective domain; ii) Identify needs; iii) Explore existing solutions; iv) List possible solutions and Identify features; v) Establish common features; vi) Recognize variable features; vii) Diagram feature model.

By analyzing the similar activities amongst these works, we were able to establish a generic SPL scoping process, illustrated in Figure 7. *Pre-Scoping* is the first task, where

supporting concepts from the scoping concept map may be used, such as metrics definition. The other activities are related which each scoping type. For instance, a market analysis may be performed in the *domain scoping* activity. As presented in the BPMN model, the activities related to 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 planning, also extracted from the concept map.

Figure 7 – A Generic SPL Scoping Process



Source: Author

4.2.1.3 Types of Scoping

Once many studies are applied to some application, organizational and economic contexts, i.e., considering different artifacts as input, the identification of scoping types is also important for the selection of an approach for SPL. In the following we present a characterization of approaches by types, deriving a conceptual map for supporting software engineers towards the selection of an approach.

Details about the scoping types alongside other information such as evaluations applied and domain of the proposal are shown in Table 7. Considering the scoping type, 18 studies covered all three types. Also, six approaches dealt with only domain scoping, eight with both asset and domain, four with domain and product, seven with asset and product scoping, and two with product scoping only. Concerning this topic, the works with higher coverage may be considered more complete. For instance, PPSMS (ALSAWALQAH; KANG; LEE, 2014) and SPLSmart (SOUZA et al., 2013) cover all scoping types and also cover several concepts as shown in Table 6.

Table 7: Studies Summary

Study	Scoping Type	Evaluation	Domain	Year
PuLSE-CDA Bayer, Muthig e Widen (2000)	Domain	Manual	SPL	1999

Table 7: Continued

Study	Scoping Type	Evaluation	Domain	Year
PuLSE deBaud e Schmid (1999)	Asset, Domain	Proof of concept	SPL	1999
PuLSE-I Bayer et al. (2000)	Asset, Domain, Product	Manual	SPL	2000
PuLSE Knauber et al. (2000)	Asset, Domain	Case Study	Architecture CAD systems	2000
Kishi, Noda e Katayama (2002)	Product	Case Study	Intelligent Transport Systems	2002
PuLSE Schmid (2000)	Domain	Case Study	SPL	2000
PuLSE Schmid (2002)	Asset, Domain, Product	Case Study	SPL	2002
PuLSE Schmid et al. (2005)	Asset, Domain, Product	Case Study	SPL	2005
Park e Kim (2005)	Domain	Proof of Concept	SPL	2005
FARE Ramachandran e Allen (2005)	Domain	Proof of concept, Manual	SPL	2005
PuLSE John et al. (2006)	Asset, Domain	Case Study	SPL	2006
Her et al. (2007)	Asset, Product	Case Study	SPL Quality	2007
Noor, Grünbacher e Briggs (2007)	Asset, Domain	Case Study	SPL reengineering	2007
Noor, Grünbacher e Hoyer (2008)	Asset, Domain	Case Study	SPL reengineering	2008
DRAMA Kim, Park e Sugumararan (2008)	Asset, Domain	Quasi-experiment, Manual	SPLE	2008
Planning Game in SPLE Carbon et al. (2008)	Asset, Domain	Case Study	Agile Methods	2008
CADSE Estublier, Dieng e Leveque (2010)	Asset, Product	Manual, Proof of concept	SPL	2010
CAVE John (2010)	Asset, Domain, Product	Case Study	SPL	2010
COPE+ Ullah, Ruhe e Garousi (2010)	Product	Proof of concept	SPL	2010
PLiCs Elsner et al. (2010)	Asset, Domain, Product	Case Study, Manual	SPL	2010
PLEvo-Scoping Villela, Dörr e John (2010)	Asset, Domain, Product	Quasi-experiment	SPL	2010
PuLSE-Eco Mærsk-Møller e Jørgensen (2010)	Asset, Domain, Product	Case Study	S-M-Sized Enterprises	2010
Cavalcanti et al. (2011)	Asset, Product	Proof of concept	SPL	2011

Table 7: Continued

Study	Scoping Type	Evaluation	Domain	Year
RiPLE-SC Balbino, Almeida e Meira (2011)	Asset, Domain, Product	Manual	Agile Methods	2011
VB Portfolio Opt. Muller (2011)	Domain, Product	Proof of concept	SPL	2011
Acher et al. (2012)	Asset, Product	Experiment	SPL	2012
Bartholdt e Becker (2012)	Asset, Domain, Product	Manual, Other	Healthcare	2012
Gillain et al. (2012)	Domain, Product	Case Study	SPL	2012
Pro-PD O’Leary, Almeida e Richardson (2012)	Asset, Product	Manual, Other	SPL	2012
RiPLE-SC Lobato et al. (2012)	Asset, Domain, Product	Case Study	Healthcare	2012
ASPLE Abbas e Andersson (2013)	Domain, Asset	Proof of concept	Self-adaptive systems	2013
Cruz et al. (2013)	Asset, Domain, Product	Proof of concept, Survey	SPL	2013
SPLSmart Souza et al. (2013)	Asset, Domain, Product	Case Study	SPL Inspection	2002
Nöbauer, Seyff e Groher (2014)	Domain	Case Study	SPL	2014
RiPLE-SC Silva et al. (2014)	Asset, Domain, Product	Case Study	Agile Methods	2014
Sierszecki et al. (2014)	Asset, Product	Case Study	Embedded controllers	2014
SPLBench Khtira, Benlarabi e Asri (2014)	Asset, Domain, Product	Proof of concept	SPL	2014
PPSMS Alsawalqah, Kang e Lee (2014)	Asset, Domain, Product	Case Study	SPL	2014
Ianzen et al. (2015)	Asset, Product	Experiment	SPL	2015
Karimpour e Ruhe (2016)	Domain, Product	Experiment	SPL	2016
Neto et al. (2016)	Domain, Product	Case Study	SPL	2016
ISPL Alam, Khan e Zafar (2017b)	Asset, Domain, Product	Manual	SPLE Security	2017
ISPL Alam, Khan e Zafar (2017a)	Asset, Domain, Product	Survey	SPLE Security	2017
CoMeS Ojeda et al. (2018)	Asset, Domain, Product	Proof of concept	SPL	2018
Small-SPL Ojeda, Rodriguez e Collazos (2019)	Domain	Quasi-experiment	S-M-Sized Enterprises	2019

Source: Author.

Riple-SC (BALBINO; ALMEIDA; MEIRA, 2011; SILVA et al., 2014; LOBATO et al., 2012) covers all scoping types by providing a well-defined agile process composed of tasks, guidelines, and roles. Another approach covering all activities, CoMeS (OJEDA et

al., 2018), presents a collaborative way of guiding the definition of the scope. The method presents tasks that a team of stakeholders may perform and will generate the represented scope of the SPL.

Another proposal covering all scoping types is PLEvo-Scoping (VILLELA; DÖRR; JOHN, 2010) which was defined as a complement and extension of other SPL scoping approaches. This goal is achieved by aiding the SPL scoping team to anticipate emerging stable features and distinguish them from unstable ones.

4.2.1.4 Adaptation

When analyzing the approaches to understand how they handle the capability of adaptation to different organizational contexts/scenarios, we can observe that the topic is covered using different strategies. Further discussion on the adaptation of existing approaches is presented next.

The work of Alsawalqah *et al.* (ALSAWALQAH; KANG; LEE, 2014) uses a set of rules for defining possible scenarios. These scenarios are responsible for the result of the cost calculations. Thus, their approach handles the scoping costs according to the scenario of the organization. Cavalcanti *et al.* (CAVALCANTI *et al.*, 2011) metamodel also considers organization scenario. The metamodel was designed for modeling the scoping according to different scenarios.

PLiCs (ELSNER *et al.*, 2010) was designed for customizable SPL. Authors discuss the approach applicability in other contexts, besides the one shown in their case study. Similar to CAVE (JOHN, 2010), which is applicable in different situations based on existing documentation, as it was designed as a generic process. COPE+ (ULLAH; RUHE; GAROUSI, 2010) also is customizable according to the organization's business and technical parameters. Her *et al.* (HER *et al.*, 2007) goes further in that area, as their framework contains guidelines to aid its application in different projects.

The proposal presented by Gillain *et al.* (GILLAIN *et al.*, 2012) is context-aware and may be instantiated in different contexts, as the authors show in their study. Their mathematical model considers market strategies as one of the aspects of defining the instantiation. Neto *et al.* (NETO *et al.*, 2016) presents a different strategy using fuzzy sets associated with a fuzzy inference system.

Lastly, PuLSE (DEBAUD; SCHMID, 1999; BAYER; MUTHIG; WIDEN, 2000; BAYER *et al.*, 2000; KNAUBER *et al.*, 2000; SCHMID, 2000; SCHMID, 2002; SCHMID *et al.*, 2005) presents a sub-process called PuLSE-BC. This sub-process defines the adaptation/customization of PuLSE-CDA according to the context where it will be applied. This customization ensures that the process and the products are appropriate. Several works have reported a customized PuLSE process (JOHN *et al.*, 2006; MÆRSK-MØLLER; JØRGENSEN, 2010; SCHMID *et al.*, 2005).

Answering RQ1: What are the similarities and differences among the approaches? We were able to define a SPL scoping concept map (Figure 6) categorizing all different concepts mapped from the studies. By analyzing the traceability of each approach to these concepts, we could understand how each concept relates to each other. This analysis gave us evidence to establish a generic SPL scoping process (Figure 7) which represents the different scoping types. For handling these required adaptations, approaches use different strategies, such as using cost models, analyzing existing documentation or deriving their approach following a specific adaptation activity.

4.2.2 RQ.2 How are existing scoping approaches evaluated?

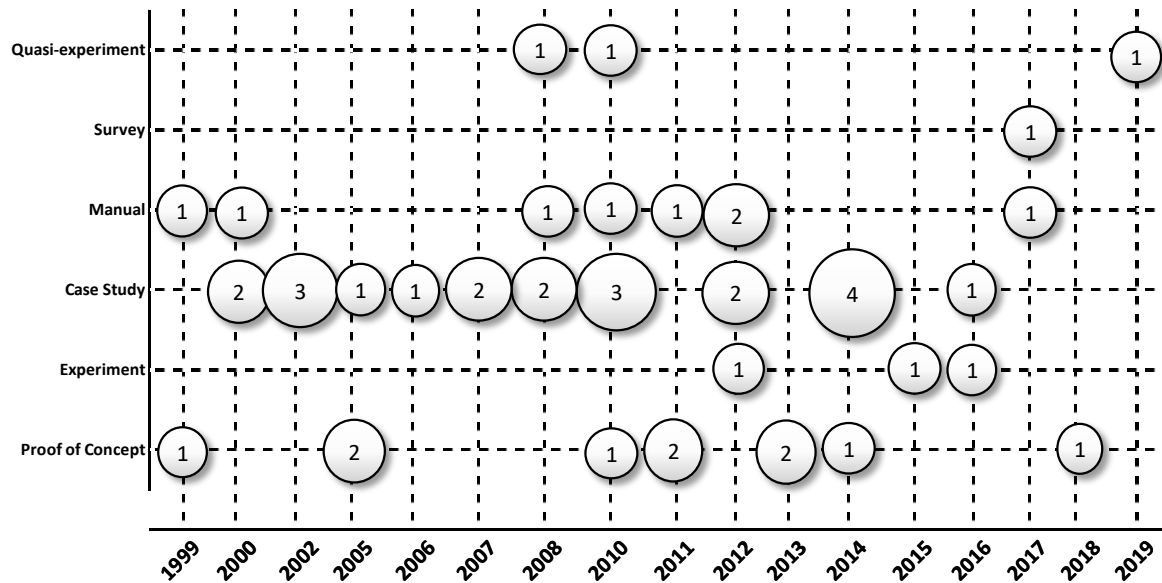
The selection of an approach may also be affected by the maturity of an approach, which is directly related to the contexts where studies have been applied. In this section, we present the methodological aspects of the analyzed studies including their evaluations and the domains where the evaluation was applied.

4.2.2.1 Evaluations Applied

As defined in our protocol, all studies included in this SLR present some evaluation, which are summarized in Table 7. In Figure 8, we crossed the evaluations with the publication year giving an overview of the number of different evaluations among the years. The most applied empirical evaluation was the case study, performed in 21 studies. The reason would be the context of a case study protocol, as we understand that applying SPL approaches in real organizations gives more reliability to the proposal. We noticed, however, a lack of controlled experiments as only three were present in the studies. Although case studies are more indicated to evaluate in real scenarios, experiments are still important when the researchers intend to monitor and compare their proposals with similar ones. Also, three works performed quasi-experiments for evaluating their approach. And one work (ALAM; KHAN; ZAFAR, 2017a) performed a survey with experts.

There were also non-empirical evaluations performed in 19 studies. The proof of concept was applied in ten studies, while manual comparisons were performed by eight, and one study performed a survey. Although these strategies for evaluating an approach may provide important results, they lack the reliability of empirical evaluations. Figure 8 also shows that the number of studies related to SPL scoping has decreased in the last five years. This decreasing number may be a result of the organization's lack of understanding of the real benefits obtained from adopting well-defined SPL scoping approaches. Such benefits include the decision-making process, discussed in Section 4.2.3.

Figure 8 – Evaluations by Year



Source: Author.

4.2.2.2 Evaluation Domains

When extracting the domain on which the approaches were evaluated, we wanted to identify if approaches were focusing on specific domains. Most of the domains identified, however, were only referred by the authors as SPL. Despite the SPL domain, we analyzed studies that were applied in systems of specific domains. ASPLE (ABBAS; ANDERSSON, 2013) may be also applied for self-adaptive systems. RiPLE-SC (BALBINO; ALMEIDA; MEIRA, 2011; SILVA et al., 2014; LOBATO et al., 2012) and the work of Carbon et al. (2008) were designed specifically for agile environments. The proposal of Kishi *et al.* was evaluated in a case study using intelligent transport systems. PuLSE was also evaluated using systems of a specific domain, the architecture CAD systems. The studies of Noor *et al.* (NOOR; GRÜNBACHER; BRIGGS, 2007; NOOR; GRÜNBACHER; HOYER, 2008) were applied for SPL reengineering of legacy systems. While ISPL (ALAM; KHAN; ZAFAR, 2017b; ALAM; KHAN; ZAFAR, 2017a) was designed for secure SPLs. Some works (BARTHOLDT; BECKER, 2012; LOBATO et al., 2012) also evaluated their approach with healthcare systems.

Considering organization scenarios, two pieces of work (MÆRSK-MØLLER; JØRGENSEN, 2010; OJEDA; RODRIGUEZ; COLLAZOS, 2019) were designed for small and medium-sized enterprises. These studies are important as usually, these companies lack the project resources of big companies for performing SPL scoping. Thus, such approaches may aid these companies to reduce the effort and costs of scoping their SPL.

Answering RQ2: How are existing scoping approaches evaluated? We conclude that most of the evaluations applied by the approaches focused on measuring how these approaches may benefit the organizations in real projects, as 21 studies applied case studies. As for evaluations domains, we identified works applicable in agile environments, self-adaptive systems, SPL reengineering, secure SPLs, and small and medium-sized companies.

4.2.3 RQ.3 How is the decision making during the process of SPL scope definition?

In this section, we discuss the topics regarding the decisions made during the approaches life-cycle. Two major aspects are used for making decisions concerning SPL scoping: cost models and metrics.

4.2.3.1 Cost Models

Costs related to scoping activities are not covered by all studies as only eight approaches presented some contribution. The work of Alsawalqah *et al.* (ALSAWALQAH; KANG; LEE, 2014) considers the estimation of feature costs which are calculated according to the scenario where the approach is being applied. Balbino *et al.* (BALBINO; ALMEIDA; MEIRA, 2011) possesses a specific task with regard to identifying business goals. During the execution of this task, economic and social activity is analyzed for identifying future costs.

Authors from (CRUZ *et al.*, 2013) formalized the inferences of development costs using mathematical equations. This is considered during the Scoping process. The work presented in (GILLAIN *et al.*, 2012) makes use of cost functions from other works. These cost functions are integrated within their mathematical model. The work of Karimpour *et al.* (KARIMPOUR; RUHE, 2016) considers profit as a goal for planning a scoping portfolio. The portfolio provides metrics such as costs for calculating profit using specific functions. Kishi *et al.* (KISHI; NODA; KATAYAMA, 2002) uses cost as a quality attribute for decision making during the scoping process. The Value-Based Portfolio Optimization approach presented in (MULLER, 2011) considers a Cost-Revenue for defining the profit related to the SPLE.

Neto *et al.* (NETO *et al.*, 2016) uses the results of feature cost metrics for generating candidate products for the SPL aiming at a higher return of investment. A different strategy is used by Noor *et al.* in their works (NOOR; GRÜNBAKER; BRIGGS, 2007) (NOOR; GRÜNBAKER; HOYER, 2008) where several strategies are used to provide initial cost/effort estimation for adapting logical components while modeling scoping.

There is also Pro-PD approach (O'LEARY; ALMEIDA; RICHARDSON, 2012)

which considers the cost calculation for taking decisions as their approach possesses cost-dependent activities. In (PARK; KIM, 2005) metrics are defined for calculating the cost of different aspects of the scoping process. While in (RAMACHANDRAN; ALLEN, 2005), cost-benefit analysis is due to mathematical expressions.

4.2.3.2 Metrics for Scoping

Considering metrics for defining scoping, in (ABBAS; ANDERSSON, 2013) the authors use quality attribute scenarios (QAS). These QAS are composed of different elements that are related to three variability questions: i) why does it vary? ii) what does vary? and iii) how does it vary? The authors also extended these QAS with additional elements than can be used for specifying the variability in terms of variation-points and constraints. Alsawalqah *et al.* (ALSAWALQAH; KANG; LEE, 2014) included in their approach the Kano model of customer satisfaction, which is used for classifying and prioritize customer needs based on how these needs affect their satisfaction.

The RiPLE-SC (BALBINO; ALMEIDA; MEIRA, 2011; SILVA *et al.*, 2014; LOBATO *et al.*, 2012) approach presents a specific phase related with metrics. During this phase, metrics are created based on business goals and are categorized as development benefits metrics and characterization metrics. These metrics are applied to prioritize a product map, selecting the features with more potential for the SPL. A similar strategy is used by Kishi *et al.* (KISHI; NODA; KATAYAMA, 2002), where the metrics are defined using a decision-making method, the analytic hierarchy process (AHP). The AHP method considers the most desirable architectural candidates according to some decision criteria. The results of this selection are used for calculating the applicability of the architectural candidates for each product of the SPL.

The metamodel presented by Cavalcanti *et al.* (CAVALCANTI *et al.*, 2011) considers the importance of metrics in SPL scoping. The metamodel includes the abstraction of these metrics allowing their instantiation based on a specific scenario. As their approach considers metrics in a modeling level, the work of Neto *et al.* (NETO *et al.*, 2016) uses metrics in a source code level. The first metric is related to the level of interdependence between scoping assets. The second metric represents the number of lines of code associated with each asset. The last metric considered is the number of flows represented by each asset of the SPL. The studies of Noor *et al.* (NOOR; GRÜNBAKER; BRIGGS, 2007; NOOR; GRÜNBAKER; HOYER, 2008) also consider metrics in a source code level such as the size of files and methods of a file. Other metrics such as complexity, dependencies, and understandability of the source code are also measured for decision making regarding the SPL.

In (CRUZ *et al.*, 2013), the cyclomatic complexity and the size of product assets are calculated considering metrics from a Metric tool. The work presented in (KHTIRA; BENLARABI; ASRI, 2014), considers metrics for performing the bench-marking of the

SPL. These metrics are the number of products per requirements, requirements per product, number of requirements per product, an average of requirements in products, requirements of the best product, new requirements, percentage of new requirements, non-implemented requirements, and percentage of non-implemented requirements.

The work of Her *et al.* (HER *et al.*, 2007) is completely designed around quality metrics. These metrics are defined based on quality attributes from the ISO/IEC 9126. As their set of metrics is extensive, three of these are specifically designed for scoping: functional coverage, non-functional commonality, and variability richness. In addition, the work of (PARK; KIM, 2005) merges metrics with cost models, using a set of metrics to analyze the economical aspects and value of the SPL. Lastly, the COPE+ approach (ULLAH; RUHE; GAROUSI, 2010) uses a small set of metrics to measure similarities among product portfolios.

Answering RQ3: How is the decision making during the process of SPL scope definition? *The decision making during the SPL scoping process is directly related to the economic aspects of the organization. Such aspects were analyzed by different studies using different types of mathematical/cost models. In this sense, cost models were used for calculating and making decisions when considering the business aspects of the organization. A similar strategy is followed by other studies; however, they have defined metrics based on business and development aspects. Such metrics are used similarly to the cost models, making decisions during the SPL development.*

4.2.4 RQ.4. What are the open research gaps and opportunities for new studies on the topic of SPL scoping?

In this section, we present and discuss the research gaps and opportunities identified by our findings. These opportunities are categorized according to SPL scoping concepts.

Decision Making: As we identified, many aspects of SPL scoping are used for decision making. In this context, some works planned to further investigate how decisions may be addressed based on different elements of the SPL. In (ALSAWALQAH; KANG; LEE, 2014), the authors discuss that business factors must be addressed to provide more comprehensive decision support for their framework. These factors may aid the team to decide how to prioritize the SPL features. In (BAYER *et al.*, 2000), the authors stated that they will work on a guidebook, containing lessons learned from their experience. Such a guidebook will be incorporated into their process. A similar gap is identified in (ESTUBLIER; DIENG; LEVEQUE, 2010), where the authors argue that heuristics could be used to improve their process. In (JOHN *et al.*, 2006), authors cite that in the future, they plan to create a model for scoping and SPL goals that may be integrated into their

approach and used for introducing the SPL concept into an organization.

Similar to these examples, in (GILLAIN et al., 2012), the authors plan to investigate in future work how to integrate customer decisions based on the analysis of competitors' products. Their work is also one of those that utilize cost models for making decisions. As the market analysis is important, so it is the consideration of cost-benefit from the SPL perspective. Thus, mathematical models capable of calculating costs are important when considering SPL scoping such as those presented in their work (GILLAIN et al., 2012). In this sense, guidelines are important to ease the complexity of this calculation. Also, the authors of (GILLAIN et al., 2012) intend to extend their cost functions to further integrate them into mathematical models. Thus, integrating risk management based on these cost models. Risk management is also discussed in the future work of (LOBATO et al., 2012). In their case, however, the idea is to combine their experience obtained for executing different case studies. Another work that mentions the improvement of cost modes as future work is (SCHMID, 2002), as authors intend to improve market aspects of their proposal.

Another aspect of decision making is the use of metrics. When analyzing the metrics proposed or used by the studies, we noticed that this aspect of the approaches needs more formalization. As reported in (CAVALCANTI et al., 2011), the management of metrics is important for detailing SPL scoping on a technical level. Using the ISO/IEC for defining such metrics may aid to formalize this process as presented in (HER et al., 2007). In this sense, the authors from (ACHER et al., 2012) plan, in future work, to adapt existing metrics to further characterize properties of the FMs generated by their approach. Another future work discussing the use of metrics is presented in (CAVALCANTI et al., 2011), where the authors plan to extend their metamodel for supporting metrics management. In (KHTIRA; BENLARABI; ASRI, 2014), the use of metrics is also cited as future work. In this case, the authors plan to add new metrics related to cost estimation, which related to the use of cost models.

Adaptation and Evaluation: Although different adaption strategies are used in different approaches, this is still a challenge, as mentioned in (SILVA et al., 2014). According to them, the SPL scoping process should consider the organizational aspects of the company. A similar problem is discussed in (KNAUBER et al., 2000), where authors argued that in small companies, where project resources and data are short, the team would not devote their time for gathering additional data to perform Scoping. However, the authors also stated that their proposal, PuLSE-Eco, handles this problem by only requiring little effort on the side of the company.

As we presented in Figure 8, empirical evaluations have being performed in most studies. However, we also identified that several authors argued that their proposals needed additional evaluations to collect more evidence about their capabilities and maturity (ACHER et al., 2012; ALAM; KHAN; ZAFAR, 2017b; BALBINO; ALMEIDA;

MEIRA, 2011; CARBON *et al.*, 2008; CRUZ *et al.*, 2013; SILVA *et al.*, 2014; ESTUBLIER; DIENG; LEVEQUE, 2010; HER *et al.*, 2007; IANZEN *et al.*, 2015; KIM; PARK; SUGUMARAN, 2008; MÆRSK-MØLLER; JØRGENSEN, 2010; MULLER, 2011; NETO *et al.*, 2016; NOOR; GRÜNBACHER; HOYER, 2008; SCHMID, 2000; VILLELA; DÖRR; JOHN, 2010; SOUZA *et al.*, 2013; SIERSZECKI *et al.*, 2014; OJEDA; RODRIGUEZ; COLLAZOS, 2019). The most common problem stated is related to the lack of sufficient evidence to consider their proposal to be reliable. Thus, many approaches still need more solid evaluations.

Other research opportunities: When analyzing the open research opportunities in the field, one aspect that was clear in the works is the lack of supporting tools. We conclude this as only a few tools were found in the proposals. For instance, DRAMA (KIM; PARK; SUGUMARAN, 2008) is a process with an automation supporting tool. Other processes use tools for specific tasks, such as PuLSE-BEAT in (JOHN *et al.*, 2006) and an untitled tool in (SILVA *et al.*, 2014). We notice, however, that tools are cited as future works of several studies (GILLAIN *et al.*, 2012; DEBAUD; SCHMID, 1999; NOOR; GRÜNBACHER; HOYER, 2008; ACHER *et al.*, 2012; ALSAWALQAH; KANG; LEE, 2014; JOHN, 2010; KHTIRA; BENLARABI; ASRI, 2014; PARK; KIM, 2005). This may indicate that developing and evaluating such tools in different organizational aspects is still an open research opportunity in the field. Hence, researchers may guide their future works by trying to answer the following RQs: **What features should be present in an SPL scoping supporting tool? How to evaluate such tool?**

A similar conclusion was achieved based on the domains for which the proposals were proposed. To the best of our knowledge, SPL reengineering is important as we understand that it is a common strategy used by organizations when migrating to an SPL context SPL (KRUEGER, 2001). In this sense, only Noor *et al.* (NOOR; GRÜNBACHER; BRIGGS, 2007; NOOR; GRÜNBACHER; HOYER, 2008) have covered this aspect, leaving opportunities for further investigation in this area when considering SPL scoping. Thus, future works may try to answer the following RQs: **How can SPL reengineering and SPL scoping be integrated? Which are the main benefits and drawbacks?**

Another aspect discussed among the works is the effort required to perform scoping related activities as this process demands much effort from the companies. In this sense, (SILVA *et al.*, 2014) argued that reducing the effort of SPL scoping is an important challenge in the field. A similar conclusion is presented in (KARIMPOUR; RUHE, 2016), where authors stated that for their approach effort should be reduced. Similar future work is stated by the authors in (ULLAH; RUHE; GAROUSI, 2010). One of the possible strategies for overcoming this challenge, as mentioned in (SILVA *et al.*, 2014), is the use of agile practices for requirements engineering. This strategy is being used in the Ripple-SC framework (BALBINO; ALMEIDA; MEIRA, 2011). However, additional strategies may

be used for handling the issue related to the demanding effort of scoping the SPL. These possibilities make a possible RQ emerges: **Which strategies and techniques may be used for reducing the effort of scoping an SPL?**

Answering RQ4: What are the open research gaps and opportunities for new studies on the topic of SPL scoping? As a result of answering our RQs, we identified several aspects of improvement in the SPL scoping field. Decision making using cost models and metrics is still not well formalized, further investigation should be performed and even guidelines may be proposed. Adaptation of approaches and their evaluation are aspects that may also require further investigation. For the latter, evaluations considering different organizational aspects may collect evidence about how the adaptation may benefit companies. Considering open research opportunities, we concluded that supporting tools, SPL reengineering, and strategies for reducing the SPL scoping effort may be investigated and even combined to increase the reliability and appeal of approaches in the field.

4.2.5 Threats to Validity

In this section, we discuss the main threats to validity related to our SLR and present how we mitigated them based on (WOHLIN et al., 2012; AMPATZOGLOU et al., 2019).

Conclusion validity: Researchers are usually not aware of their own bias during the analysis and classification of studies. This bias could negatively impact the results of the SLR. An additional threat is a possible inaccuracy during data extraction. Therefore, trying to mitigate both threats, two researchers independently performed the QA and data extraction from the studies. In the case of divergences, an additional researcher was assigned to discussing divergent points. Lastly, the fishing and error rate problems may impact the conclusions of an SLR. We handle this problem by achieving our conclusions and answering our RQs after collecting and analyzing the results of all 45 studies.

Internal validity: Publication bias refers to SPL scoping approaches that were not selected due to research results not being satisfactory. To mitigate this threat, the analysis of the studies in this field was performed considering a large sample of the studies. An additional threat is the inclusion of studies considered to have low quality, thus, negatively impacting the RQs answers. For mitigating this threat, we defined and applied QAs, classifying, and guaranteeing that each study selected contained at least the minimum information required for answering our RQs. Another possible threat is the low number of studies analyzed, which may not represent the field. We mitigated this problem by selecting and analyzing a large set of studies. When compared to related reviews (see Section 4.3), we analyzed 30 additional studies.

Construct validity: Not retrieving studies due to their absence from a certain

database is an additional major threat. We mitigated this problem by using different digital libraries which index a large amount of conferences proceedings and journals. Besides, we conducted a snowballing, which is a technique independent of DLs. To mitigate the issue related to the inclusion/exclusion of relevant works, two researchers applied independently a set of well-defined exclusion and inclusion criteria, as well as QA for qualification and classification of studies. In the case of divergences, a third researcher would present its opinion. These strategies give our SLR reliability regarding the studies retrieved.

External validity: A possible external threat is related to the coherence of our results. In this sense, as we based our SLR protocol on well-defined SLR guidelines (KITCHENHAM et al., 2010) we believe that we achieved satisfactory results for answering our RQs. Considering the RQs answers, we tried to perform a generic analysis of the data collected, considering both academic and practice points of view. An additional threat is related to the incorrect identification of the research opportunities discussed in RQ4. To mitigate this threat, we extracted the limitations and future work described by the authors in each study. After documenting this information, we analyzed whether the limitations of one study was covered by another or not. We also considered open research opportunities that were pointed out as future work by several studies, such as the development of supporting tools.

4.3 Related Work

When considering secondary studies, we may find several in the SPL field. For the past decade, more than 60 relevant secondary studies, including systematic reviews were conducted in this field (MARIMUTHU; CHANDRASEKARAN, 2017). Among these, we have reviews focusing on SPL requirements (ALVES et al., 2010; SEPÚLVEDA; CRAVERO; CACHERO, 2016; KHURUM; GORSCHER, 2009), SPL quality attributes (MONTAGUD; ABRAHÃO; INSFRAN, 2012), SPL reengineering (ASSUNÇÃO et al., 2017; LAGUNA; CRESPO, 2013) and SPL testing (MACHADO et al., 2014; ENGSTRÖM; RUNESON, 2011; NETO et al., 2011). Despite their important contributions to the field, however, there are few studies in the literature comparing SPL scoping approaches. Table 8 summarizes these contributions: three studies identified as related to this SLR. In this section, we describe these studies in comparison with our SLR, discussing their focus in analyzing the state of the art for SPL scoping approaches.

Schmid (SCHMID, 2000) reported a survey analyzing a set of scoping approaches. The survey analyzed technological approaches considering four dimensions: scoping tasks, the object of scoping, scoping product, and scoping process. The survey considered works outside the software discipline. The author used a framework to structure the scoping field with four goals: organize and structure the scoping field; analyze scoping approaches considering their benefits and drawbacks to provide an overview of the field; assist the selection of existing approaches; assist the improvement of existing methods and de-

Table 8 – Summary of Related Work

Ref.	Year	Protocol	Studies Analyzed	Main Goal
Schmid (2000)	2000	Survey	13	Analyze technological proposals for SPL scoping considering scoping tasks, object of scoping, scoping products and scoping process.
John e Eisenbarth (2009)	2009	Survey	16	Investigate SPL scoping approaches among their goals, variability management; inputs and outputs among other characteristics.
Moraes, Almeida e Romero (2009)	2009	SLR	13	Investigate SPL scoping approaches and identify scope definition techniques, analyzing their properties, in addition to strong points and drawbacks.
Our SLR	2019	SLR	45	Identify similarities and differences among SPL scoping approaches and processes, business aspects, conceptual characteristics and research opportunities.

Source: Author.

velopment of new ones. The approaches analyzed were categorized into three scoping categories: product line scoping, domain scoping, and asset scoping. When conducting our SLR, we apply the same classification. This classification was combined with the goal of the approaches which could be for identification, evaluation, or optimization of scope.

The survey results may be used to better understand the SPL scoping activity, by organizing its results among the proposed dimensions it was possible to identify considerable differences among the proposals. Although we have similar goals, our SLR will focus on more technical aspects of the proposals, such as scoping types, metrics, and cost models.

John e Eisenbarth (2009) presented the results of another survey aiming to investigate some aspects in SPL scoping: the goal of the approaches; variability management; inputs and outputs of the approaches; roles; effort to perform scoping activities; and maturity and benefits of the approaches analyzed. To investigate these aspects the authors formulated three main goals: identify the connection between scoping and requirements engineering; identify the connection between scoping and architecture; how the approaches handle the production of quantifiable results. These two works (SCHMID, 2000; JOHN; EISENBARTH, 2009), did not apply any systematic review protocol, thus their reviews are not repeatable. The SR presented in (MORAES; ALMEIDA; ROMERO, 2009), however, applied such a protocol allowing users to reuse its protocol.

The main goal of the SR presented by Moraes, Almeida e Romero (2009) is to investigate the existent SPL scoping approaches to identify scope definition techniques, analyzing their properties, in addition to strong points and drawbacks. Their strongest contribution is to serve as a guide to practitioners to identify the more appropriate approach to be used in an industrial context. These goals are similar to our SLR goals. However, we intend to identify information that was not analyzed by them, such as

re-factoring strategies, commonalities and variabilities, and domains. Despite these differences, as their work was conducted in 2009, our SLR included SPL scoping approaches published in the last ten years. Their work reported the result of an analysis performed in eleven primary studies, which were used to extract information used in our protocol, such as keywords, and terms for the search strings definition.

Another comparison work was presented in (LEE; KANG; LEE, 2010) where the authors compared and analyzed three called “mainstream approaches”. The authors presented a framework to perform such comparisons. The result was the extraction of their essential components and the creation of a unified approach.

4.4 Chapter Lessons

In this chapter, we presented the protocol and results of a SLR on SPL Scoping. With the results of this SLR, we could achieve different contributions:

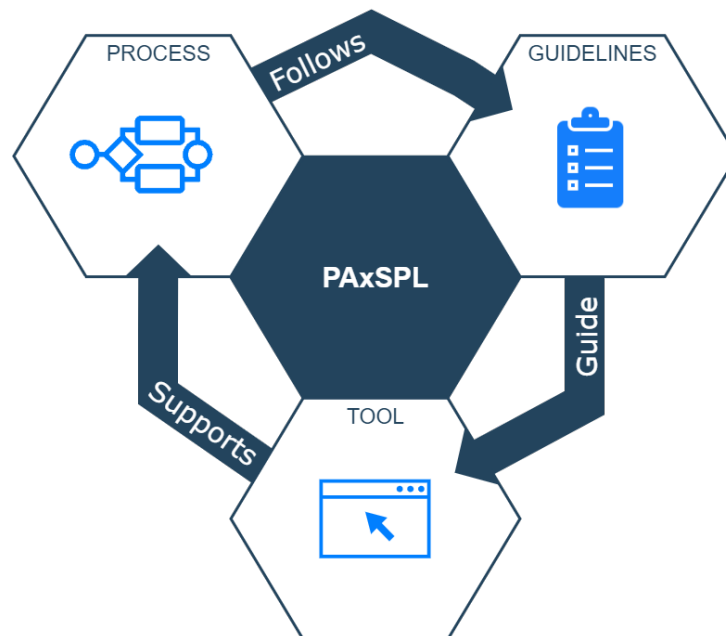
- i We provide a comprehensive analysis of the current literature on the topic of SPL scoping. Our work encompasses more than 10 years of advance in this research topic with 32 additional papers about previous mappings and surveys on this topic.
- ii For practitioners our work contributes by providing a generic scoping process, based on similarities found on existing approaches, to guide those companies envisaging the adoption or migration towards SPLs.
- iii This SLR supports researchers on understanding the current body of knowledge on SPL scoping in terms of existing approaches, concepts. Furthermore, we describe identified open challenges and research opportunities to conduct new studies.

This SLR is important to our work due to two aspects: i) it provides information used form improving PAXSPL; ii) it is used for identifying related works, discussed in Chapter 7.

5 PAXSPL

In this chapter, we present the PAXSPL framework. The version presented here includes the improvements made based on the future work defined in Marchezan et al. (2019b). Section 5.1 presents an overview of the PAXSPL process, detailing each phase. Section 5.2 discusses how the customization is handled in terms of feature retrieval and SPL scoping. The PAXSPL guidelines are presented in Section 5.3. The supporting tools are discussed in Section 5.4. These two aspects are part of our proposed framework. This framework is presented in Figure 9. As stated, the framework is composed of a process that follows a set of guidelines that guide the tool execution, supporting the process.

Figure 9 – PAXSPL Framework



Source: Author

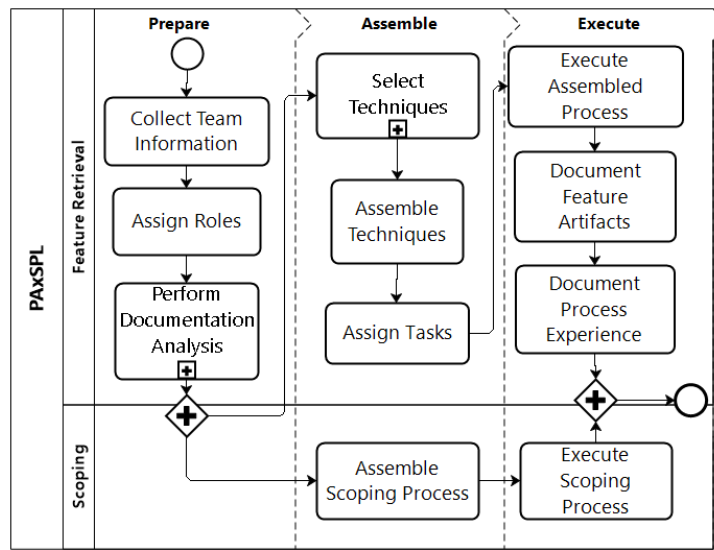
5.1 PAXSPL Process

To aid the decisions related to selecting the strategies and techniques for feature retrieval, and SPL scoping, we defined PAXSPL main workflow. The process is presented in Figure 10, divided into three main phases: prepare, assemble, and execute.

5.1.1 Prepare

To prepare the process assembly, the information is collected in the first phase of PAXSPL. The first activity is, therefore, to **Collect Team Information**, as illustrates Figure 10. During this activity, information about the team is collected. This information

Figure 10 – The Prepare, Assemble and Execute Process for Software Product Line Reengineering.



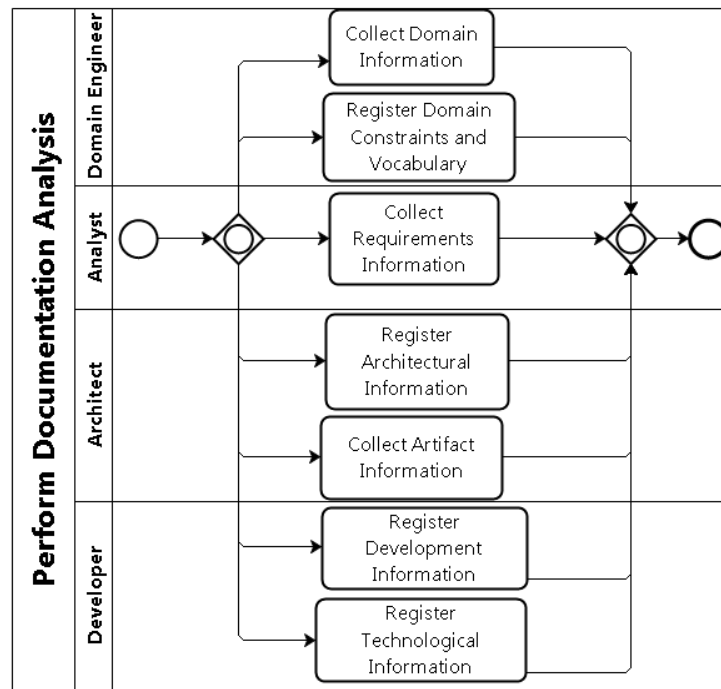
Source: Author.

includes the experience, skills, knowledge, and preferences of each member. The second activity is **Assign Roles** based on the information collected on the previous activity. Possible roles are: Domain Engineer, Analyst, Architect and Developer. These roles are related with the following sub-process, which is **Perform Documentation Analysis**, detailed in Figure 11. Here, domain information, constraints, and glossary are collected by the **Domain Engineer**. This activity allows the user to collect and analyze information regarding SPL scoping, such as descriptions of domain models and reuse benefits and risks. Requirements information of the products is gathered by the **Analyst**. Information about architecture and artifacts is registered by the **Architect**. Also, the **Developer** may document technologies and development information. The main contribution of **Prepare** is the generation of a **Documentation Set**, composed of artifacts and information used during the **Assemble** phase. The **Documentation Set** artifacts may include product architecture, requirements, domain information, and team information. Some artifacts have a higher level of impact when choosing a technique, but they all must be used to assemble the process.

5.1.2 Assemble

In this phase, the information collected is analyzed and techniques are selected and assembled into the generic process. The first sub-process is to **Select Techniques**, here, the data collected previously is analyzed to help the selection of techniques for feature retrieval (Figure 12). First, a candidate technique is selected to be analyzed. This selection is made based on artifacts from the documentation set, (*e.g.*, team members experience with the technique). After the candidate technique is selected, it must be

Figure 11 – Perform Documentation Analysis Sub-process.

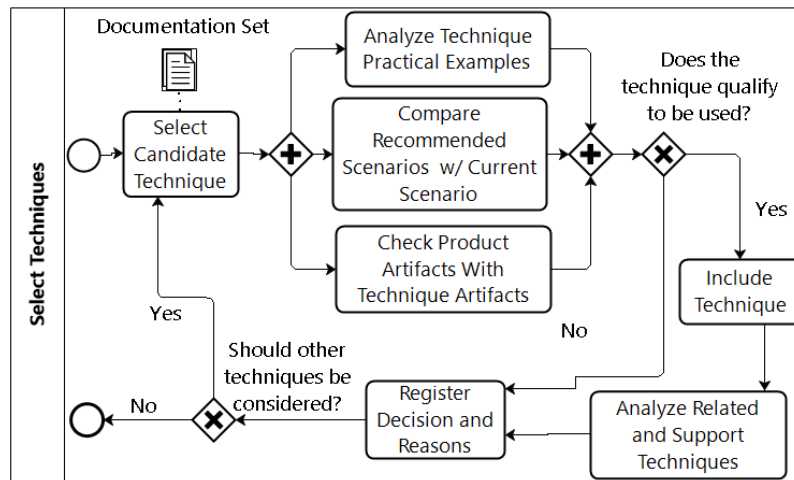


Source: Author.

analyzed considering three points. The first is practical examples, which are examples of the technique found in the literature. The second is a comparison among techniques recommended scenarios with the current scenario, also found in the literature. The users should analyze whether their scenario contains some similarities in comparison with the scenarios for which the technique was used. Lastly, users should check available product artifacts with technique artifacts, for instance, if a technique uses requirements artifacts, this type of artifact should be available. Based on the analysis performed considering those three points, the user must decide to use the techniques or not. If the technique qualifies to be used, it is included as a selected technique, then related and support techniques are analyzed to decide whether they may be candidate techniques. The decision of including or not the technique must be registered along with its reasons. If another technique is considered a candidate the process repeats.

The second activity during the Assemble phase is **Assemble Techniques** (Figure 10). In this activity, the chosen techniques are assembled inside our generic process, shown in Figure 13. The generic process consists of the main activities performed during the feature retrieval process considering the literature mapped during our process creation. **Extract**, **Categorize** and **Group** are basic activities performed with the features prior to the creation of the feature model. After the end of each activity, we placed a check gateway, implying to the user that some kind of checking may be performed before moving to the next activity. The retrieval techniques, presented in Figure 15 are assembled into the generic process, generating the assembled process. The assembled process is

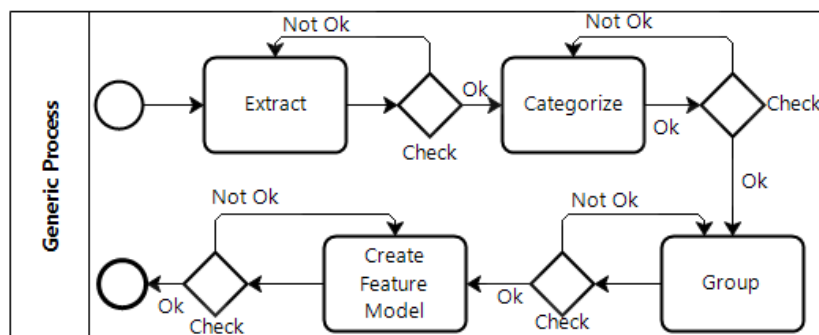
Figure 12 – Select Techniques Sub-process.



Source: Author.

customized according to the scenario where PAXSPL has been applied, giving our process flexibility to be applied in different situations. To help the assembly of techniques, we created in the guidelines what we call a priority order attribute for each technique. This attribute shows in which step of our generic process each feature retrieval technique may best fit. The last **Assemble** activity is **Assign Tasks** where each member of the team will receive a task to perform during the retrieval process execution. These tasks are not directly related with the roles used during the **Perform Documentation Analysis** activity, they are related to the retrieval tasks (*e.g.*, extract features). In this case, one member may perform more than one task and a task can be performed by multiple members.

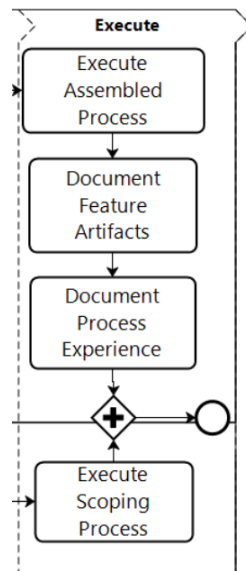
Figure 13 – The Generic Process for Feature Retrieval and Analysis.



Source: Author.

Considering the scoping line pool in Figure 10, we have a parallel gateway which divides the main workflow into the feature retrieval and scoping. Thus, still, during the **Assemble** phase, the scoping process should be assembled using the scoping concept map (see Figure 17) and the scoping generic process (see Figure 18).

Figure 14 – PAXSPL Execute Phase



Source: Author.

5.1.3 Execute

During this phase, see Figure 14, 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 others. In parallel to these activities, the scoping process is being executed, scoping artifacts may be traced with feature artifacts. 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-execution 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.

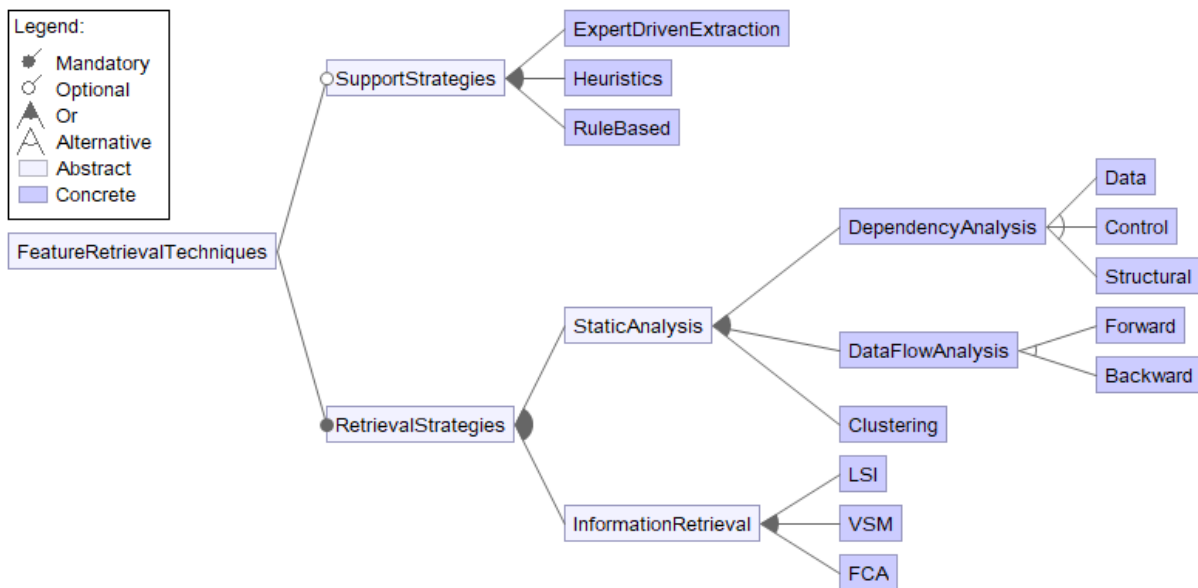
5.2 Customization for Different Scenarios

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 and SPL scoping.

5.2.1 Customization for Feature Retrieval

As illustrated in Figure 15, we grouped the techniques based on their strategy. We have the mandatory group of Retrieval Techniques which are composed of two Or-alternatives (at least one must be selected (CZARNECKI; EISENECKER, 1999)), Static Analysis, and Information Retrieval techniques.

Figure 15 – A Feature Model of Retrieval Techniques.

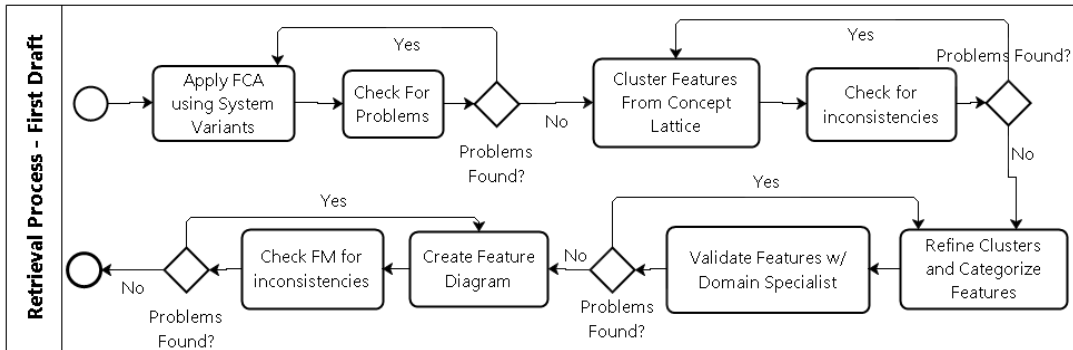


Source: Author.

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: LSI, VSM (ALVES et al., 2008) and FCA. The second group is optional, composed of 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 and analysis, shown in Figure 13. As stated, the generic process is used to tailor the assembled process that generates the extracted features and the feature model. We give an example of the assembled process in Figure 16, in which FCA and clustering can be assembled as extraction techniques to retrieve the features and create the Feature Model. In this particular case, the tasks were not assigned to a specific actor, which means they may be performed by anyone. As Figure 16 shows, a concept lattice is checked to find problems. Then, the extracted features are grouped into clusters and a check for inconsistencies is performed. The clusters are refined, the features are categorized and validated with the domain specialist. Lastly, the feature model is created and checked, and the process ends.

Figure 16 – An Assembled Process

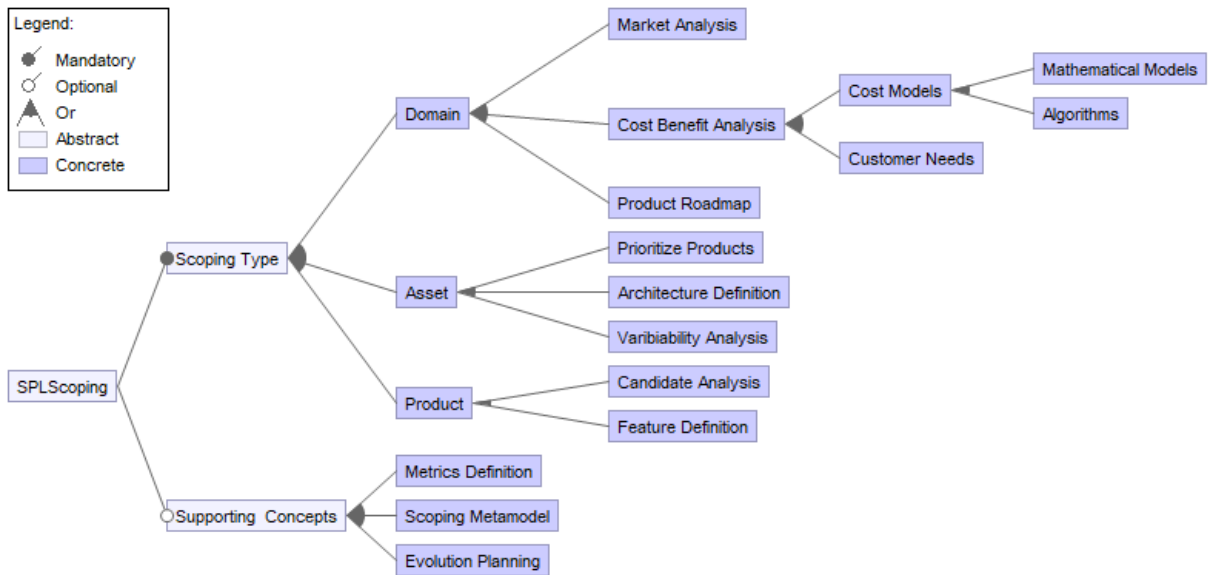


Source: Author.

5.2.2 Customization for SPL Scoping

In addition to the feature retrieval, PAXSPL also guides customization considering the Scope of the SPL. By analyzing a set of 45 works citing SPL scoping proposals (see Chapter 4), we were also able to establish a feature model of Scoping activities and concepts. Figure 17 presents these activities which are divided by Scoping type and Supporting concepts.

Figure 17 – A feature Diagram of SPL Scoping Activities



Source: Author

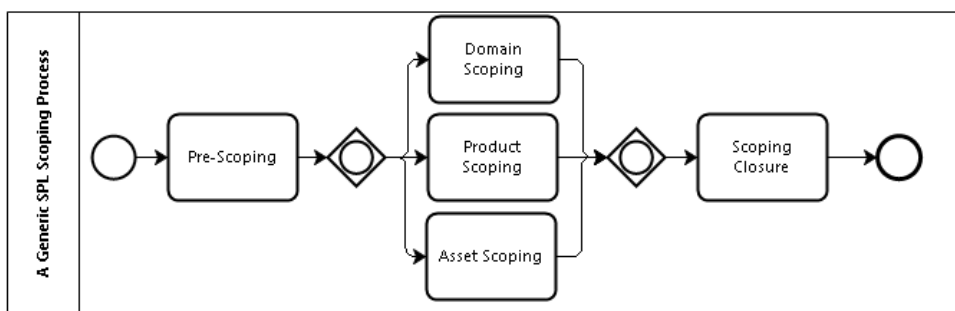
All features in the model are Or-alternative, except by the supporting concepts, which are optional. Among the supporting, we have the definition of metrics for scoping, use, or definition of meta-models and the SPL evolution plan. Considering the scoping types, they are three: Domain, Asset, and Product. A domain is subdivided into market analysis, cost-benefit analysis, and product roadmap. Continuing, the asset scoping

may be divided into prioritizing products, the 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 users and assembled into a generic SPL Scoping process, presented in Figure 18. *Pre-Scoping* is the first task, where supporting concepts from the Scoping feature model may be used, such as metrics definition.

The users would then assemble the activities related to which scoping type. The selection of activities is performed based on the user's context. As presented in the Business Process Model and Notation (BPMN) model (Figure 18), the activities related to 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. To better represent this process, we present an example of the scoping process assembled in Figure 19. The process starts by defining metrics based on business aspects, then deriving these metrics for specific projects. Both of these activities are part of the pre-scoping. After the pre-scoping is finished, the candidate features are analyzed and their names and descriptions are defined, these two activities are examples of product scope. While these are executed, the market is analyzed in an activity related to domain scoping. Lastly, a plan for evolving the SPL is defined in the scoping closure.

Figure 18 – A Generic SPL Scoping Process.

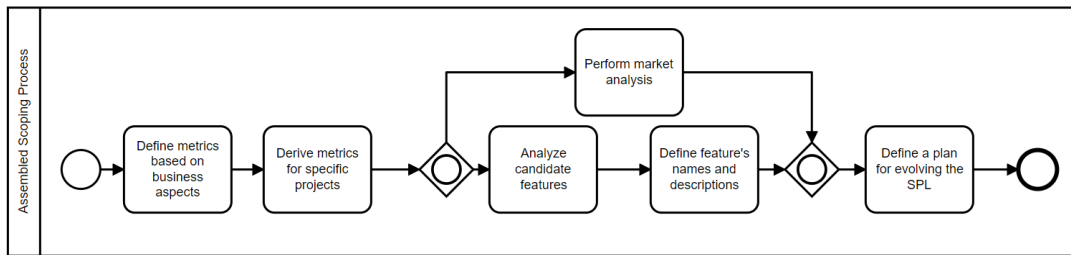


Source: Author

5.3 Guidelines

In this section, we discuss the second part of the PAXSPL framework, a set of guidelines. These guidelines were created to present retrieval techniques strategies, describe each technique, give examples, supporting tools, define recommended scenarios, and give a prioritization assemble order when assembling a technique into the generic process. In our guidelines we also included basic information about SPL, SPL Scoping, SPL reengineering, variability management, and feature model notations. This information

Figure 19 – An Assembled SPL Scoping Process.



Source: Author

is important for users with low experience with SPL reengineering and may help mitigate their difficulty when performing PAXSPL. Within our guidelines, we also included a checklist (see Section 5.3.1) to give support during the process execution.

With this kind of documentation, we intend to give as much help as possible for PAXSPL users when selecting the techniques for feature retrieval. We also created a feature model of Feature Retrieval Techniques that formalizes possible techniques and makes it possible to calculate how many different combinations can be chosen. These techniques are those illustrated in Figure 15.

In addition to the feature retrieval techniques, we added for PAXSPL new version the SPL scoping guidelines¹. These contain information with regard to the concepts/activities presented in Figure 17.

5.3.1 Support Checklist

The guidelines contain a support checklist to be used by users. This checklist was created to give support during PAXSPL execution. It contains items for each activity of our process and may reduce the difficulty to conduct the process and to make some decisions as well. All questions of the checklist should be answered with *yes* or *no*. The checklist is described as follows:

1. Prepare Phase

- 1.1. Is the team information (members skills, experience, roles) registered?
- 1.2. Does the company possess a business organization chart?
- 1.3. Is the team information registered using a template?
- 1.4. Are all the roles assigned to team members?
- 1.5. Do all team members possess at least one role?
- 1.6. Are the assigned roles related to team members' experience, skills, or role in the company?

¹ The full guidelines documentation is available at <<https://github.com/HestiaProject/PAXSPL/wiki/Guidelines#spl-scoping-concepts>>

1.7. Are the Feature Retriever and Feature Tester roles assigned to different team members?

1.8. Documentation Analysis

1.8.1. Do the system variants possess important domain information?

1.8.2. Do the system variants need a domain glossary?

1.8.3. Do the system variants need a domain constraints list?

1.8.4. Are the requirements well documented?

1.8.5. Is the architectural information well documented?

1.8.6. Is the artifacts type information registered in a document?

1.8.7. Is the development and technological information (programming patterns, programming, and development paradigms) registered?

2. Assemble Phase

2.1. Select Techniques

2.1.1. Is the candidate technique related to some team member skills or experience?

2.1.2. Is the candidate technique related to other selected technique?

2.1.3. Were the technique practical examples analyzed?

2.1.4. Are the practical examples somehow similar to the current scenario(s)?

2.1.5. Is the candidate technique recommended scenarios similar to the current scenario(s)?

2.1.6. Are the candidate technique input artifacts available from the system variants artifacts?

2.1.7. Does the candidate technique have related techniques?

2.1.8. Is a support technique needed?

2.2. Assemble Techniques

2.2.1. Is the technique priority to be assembled for extraction?

2.2.2. Is the technique priority to be assembled for categorization?

2.2.3. Is the technique priority to be assembled for group?

2.2.4. Can the techniques be combined for one activity (extract, categorize, or group)?

2.2.5. Are there techniques assembled for the three first activities (extract, categorize, or group) of the generic process?

2.3. Are all activities of the assembled process assigned to at least one team member?

2.4. Were the scoping activities assembled into the Scoping Generic Process?

3. Execute Phase

- 3.1. Was the assembled process executed without major problems?
- 3.2. Were the retrieved features verified by the feature tester?
- 3.3. Are the feature artifacts well documented?
- 3.4. Was a feature diagram of the retrieved features created?
- 3.5. Was the assembled scoping process fully executed?
- 3.6. Was the process preparation, assembly, and execution experience registered?

5.3.2 Retrieval Techniques Tool Support

Lastly, our guidelines contain information about tools found in the literature that may be used to perform the feature retrieval techniques. Most information was extracted from Assunção et al. (2017) SMS. Within our guidelines, users may find the tools name, basic description, link for download, citations where the tool was used, and for which retrieval techniques they may be used².

5.4 PAXSPL Tool

In this section, we present the third part of the PAXSPL framework, a supporting tool developed to support the process' life-cycle.

5.4.1 Requirements

As a result of the work presented in Marchezan et al. (2019b), where we conducted a case study to measure PAXSPL in terms of effort, we concluded that a supporting tool would reduce the effort of conducting some activities of the process. Thus, we started the development of such a tool. We defined that the tool would have four distinct actors, and nine Use Case (UC), presented in Table 9. The actor *Manager* performs the UCs: UC1. Manage Project, UC2. Manage Team, UC3. Document Process Experience. Actor *Feature Retriever* performs UC4. Execute Retrieval Process. The *Feature Tester* performs UC5. Check Features Artifacts. Lastly, *Team Member* performs UC6. Select Strategies and Techniques, UC7. Generate Documentation Set, UC8. Create Feature Model and UC9. Configure Products.

After deriving these UCs into more detailed requirements, we started the design of our tool.

² More information at <<https://github.com/HestiaProject/PAXSPL/wiki/Tool-Support>>

Table 9 – Use Cases for PAXSPL Tool

Actor	Id	Name
Manager	UC1	Manage Project
	UC2	Manage Team
	UC3	Document Process Experience
Feature Retriever	UC4	Execute Retrieval Process
Feature Tester	UC5	Check Feature Artifacts
Team Member	UC6	Select Strategies and Techniques
	UC7	Generate Documentation Set
	UC8	Create Feature Model
	UC9	Configure Products

Source: Author

5.4.2 Design

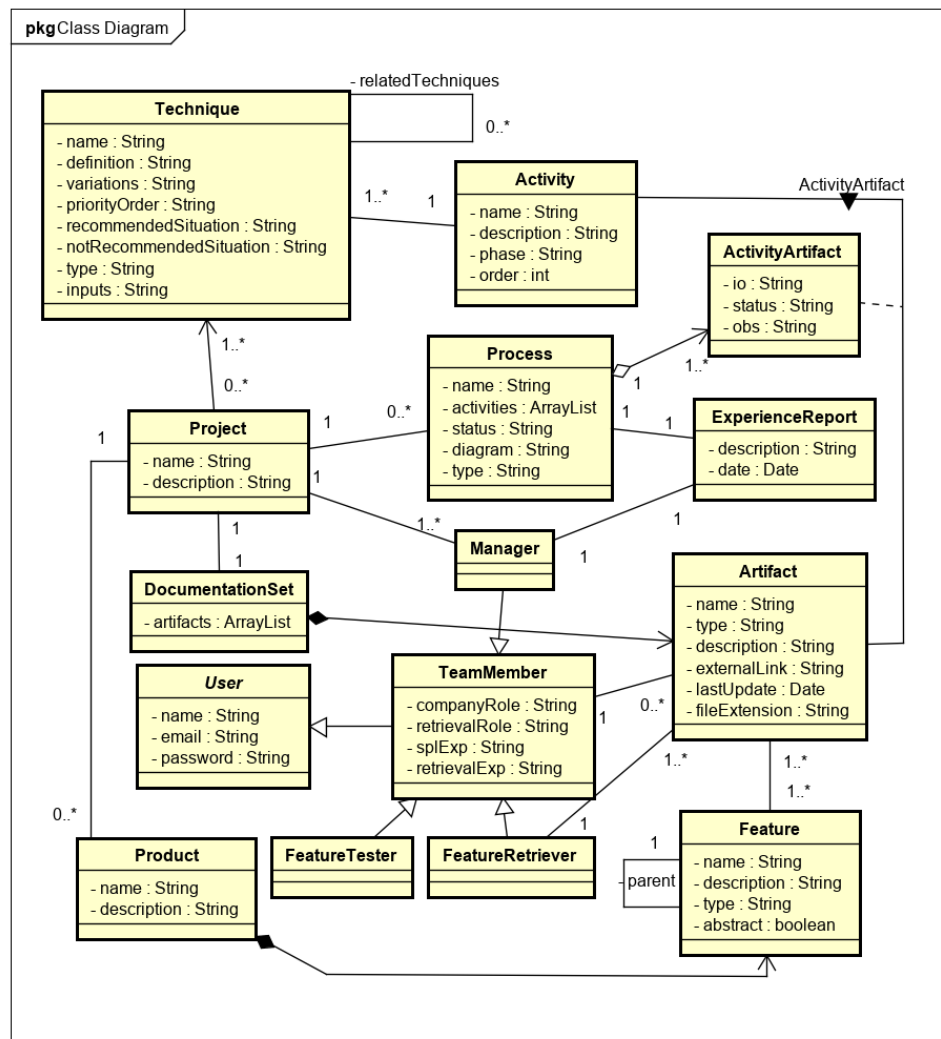
Figure 20 presents the class diagram constructed based on the requirements. We defined a *Project* to be the central part of the tool. This *Project* is managed by one *Manager*, may have several *Processes* (e.g. feature retrieval and scoping processes) and one *Documentation Set*. This *Project* also possess several *Techniques*. The *Manager* is a specialization of a *Team Member*, which is a type of *User*. The *Team Member* also generates different *Artifacts*, which are part of the *Documentation Set*. All these classes cover the first phase of PAXSPL, **Prepare**.

Continuing, the *Process* is an aggregation of *Activities*, which may have from one to several *Techniques* that may be for feature retrieval or scoping. The *ActivityArtifact* is an associative class representing an activity that possesses artifacts linked to it. Thus, the *ActivityArtifact* class has and status, observations for each artifact and an attribute called “io” which shows wheter the artifact is an input or output. These classes cover PAXSPL’s second phase, **Assemble**. Then, we have the *Features* extracted by the *Feature Retriever* and checked by the *Feature Tester*. The features may be related to several artifacts and the artifacts may be related to several features. Also, products can be generated. The *product* class is a composition of *features* and is related to a project. Lastly, the *Manager* must generate an *Experience Report*. These classes cover the last phase of PAXSPL life-cycle, **Execute**.

After defining the requirements and the class diagram, they were analyzed with the purpose to support the decision-making regarding design and development of the tool. Thus, we established Design Decision (DD) as follows:

DD1. The tool must cover all activities of the PAXSPL process: we want to centralize the process execution around the usage of the tool. Thus, the PAXSPL supporting tool will support all activities of the PAXSPL life-cycle. Also, the tool will support the activities created and assembled into the generic process by the users.

Figure 20 – PAXSPL Supporting Tool Class Diagram.



Source: Author.

- DD2. The tool must be open-source and free to use:** as an academic work, we desire that the tool may be used by researchers and practitioners of the field. Thus, collecting their feedback to improve our tool.
- DD3. The tool must allow work among multiple users:** as there are different types of users (e.g. Manager and Feature Retriever) generating and managing similar artifacts, the tool must support collaborative work.
- DD4. The tool must be developed as a web-tool:** two reasons for this are that PAXSPL aims at been executable in different organizational scenarios (including remote work), and the need for collaborative work with several users performing different tasks during the process execution.
- DD5. The tool must allow the modeling of a BPMN processes:** as one of the most important UC of our tool is the assembly of the feature retrieval process, our tool

must allow users to model their process as they desire. The BPMN representation may aid users to better visualize and understand their process as BPMN is a well-known and defined process language.

DD6. The BPMN modeling environment should be build using a JavaScript library: as BPMN is a well know language, there are many libraries available online providing resources for modeling BPMN processes. In this sense, built an in house solution for modeling such a process is not necessary.

DD7. The tool must allow the users to model a FM according to a specific notation: as FM is the most common way of representing the SPL variability, and also is a mandatory activity in our generic process, the tool should allow for the users to create their FM. This FM however, should follow the rules and structure of a specific FM notation, to be adequate to SPL variability standards.

DD8. The tool must allow the users to configure products according to the FM: in this case, the user may generate different products by selecting the features from the FM according to their rules (mandatory, optional, XOR-Alternative and OR-Alternative).

Thus, to follow DD1, we used PAXSPL life-cycle as a basis for creating the workflow and screen navigation of our tool. Based on DD2, we created a public repository³ at GitHub, then the development process may be followed by researchers and practitioners of the field. For addressing DD3 and DD4, we selected the PHP programming language and the Laravel Framework⁴ for developing our solution. For DD5 and DD6 we decided to use the *bpmn-js* toolkit⁵ which allows for editing and visualizing BPMN diagrams in any web application. The toolkit uses the BPMN 2.0 standard, which satisfies our requirements. To address DD7, we decided to use the FM notation presented in Czarnecki e Eisenecker (1999). This notation contains all the mandatory aspects of a FM as well as the optional aspects that are most used by SPL engineers (SEIDL; WINKELMANN; SCHAEFER, 2016). For DD8 we decided to integrate an open-source feature configurator⁶ to reduce the effort of developing this configuration process.

5.4.3 Running Example

In this section, we present an example of the use of the tool⁷, describing some of its screens. For this case, we considered a scenario based on a case study presented in Eyal-Salman, Seriai e Dony (2013). Figure 21 shows the project home screen, divided

³ <<https://github.com/HestiaProject/PAXSPL/tree/master/Tool/paxspl-tool>>

⁴ <<https://laravel.com/>>

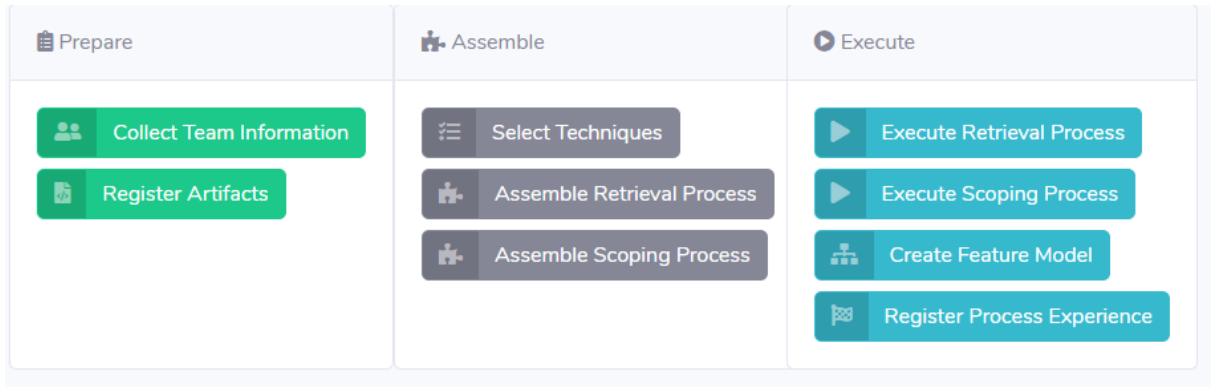
⁵ <<https://github.com/bpmn-io/bpmn-js>>

⁶ <<https://github.com/ekuiter/feature-configurator>>

⁷ The tool is currently available at <<http://paxspl-tool.herokuapp.com/>>

into the three phases of the PAXSPL life-cycle. From this screen, the user may access all activities of the life-cycle, however, the tool will only provide access to an activity (*e.g.* select techniques) if all prior activities were completed. Following the correct order, the user first will collect team information. Then, the artifacts must be registered. Figure 22 illustrates the artifacts screen. In this case, we have artifacts, where their names, types, owner, and modification dates are displayed.

Figure 21 – Project Home Screen



Source: Author

Figure 22 – Artifacts Screen

Artifacts of project: Text Editor SPL

[New Artifact +](#) [Download Artifact Report 📄](#)

Name	Type	Owner	Last Update Date	Last Update by	Action
Source Code	Development	Luciano	04-17-2020	Luciano	View 👁 Edit ✎ Remove 🗑
Domain Glossary	Domain	Luciano	02-19-2020	Luciano	View 👁 Edit ✎ Remove 🗑
Use case specification 2	Requirements	Luciano	02-29-2020	Luciano	View 👁 Edit ✎ Remove 🗑

Source: Author

Once all artifacts are registered, the user should select the retrieval techniques for the project. Figure 23 illustrates part of the techniques screen. Besides the technique name and type, it is also shown the recommendation level of each technique. This level is calculated based on the current scenario, which considers the team members' information and the artifacts of the documentation set. For instance, if a team member has experience with Formal Concept Analysis, the recommendation level of these techniques is increased. Also, if a technique uses as input an artifact which is part of the documentation set, this technique recommendation level is increased. Also, there is (at the top of the screen) the

button for downloading a report containing the description of all techniques selected and the reasons for selection.

Figure 23 – Techniques Screen

Name	Type	Recommendation	Action
Clustering	Static Analysis	31%	View Remove
Dependency Analysis	Static Analysis	31%	View
Data Flow Analysis	Static Analysis	31%	View
Formal Concept Analysis	Information Retrieval Techniques	100%	View Remove

Source: Author

The next step is to add activities to the feature retrieval and scoping generic processes. Figure 24 shows the screen where activities were included in the *Group* and *Create Feature Model* phases of the generic feature retrieval process. The activities have a name, an order of execution, and a retrieval technique to be performed. A similar screen is also present in the tool, allowing the users to include activities in the generic scoping process.

To better represent the assembled processes, our tool also allows to model a BPMN representation. Once a process is created, the tool generates a generic BPMN representation. Figure 25 shows the screen where the generic BPMN representation of the process may be customized. In this case, we have changed the generic process into a process with five activities. As defined in the DD, we used an open-source library for developing this functionality. This screen provides the users all required functionalities for modeling a BPMN process.

After the processes are assembled, the user should execute them. Figure 26 shows the screen of an activity execution, allowing the users to provide inputs and outputs. The user may click *Pause* to put that activity on hold and continue it later, or the activity may be concluded. Once all activities are finished, the users assigned as Feature Tester may check the artifacts generated. If no artifacts presented problems, the feature model may be created.

Figure 27 shows the screen where the feature model is created. The tool allows creating a feature model based on the same feature notation used by the FeatureIDE (KASTNER et al., 2009) tool. The features may be abstract, optional, XOR-

Figure 24 – Activities Screen

Group:

New Activity +

Name	Order	Retrieval Tech.	Action
Derive code-topics from common class partitions;	1	Clustering	View Edit

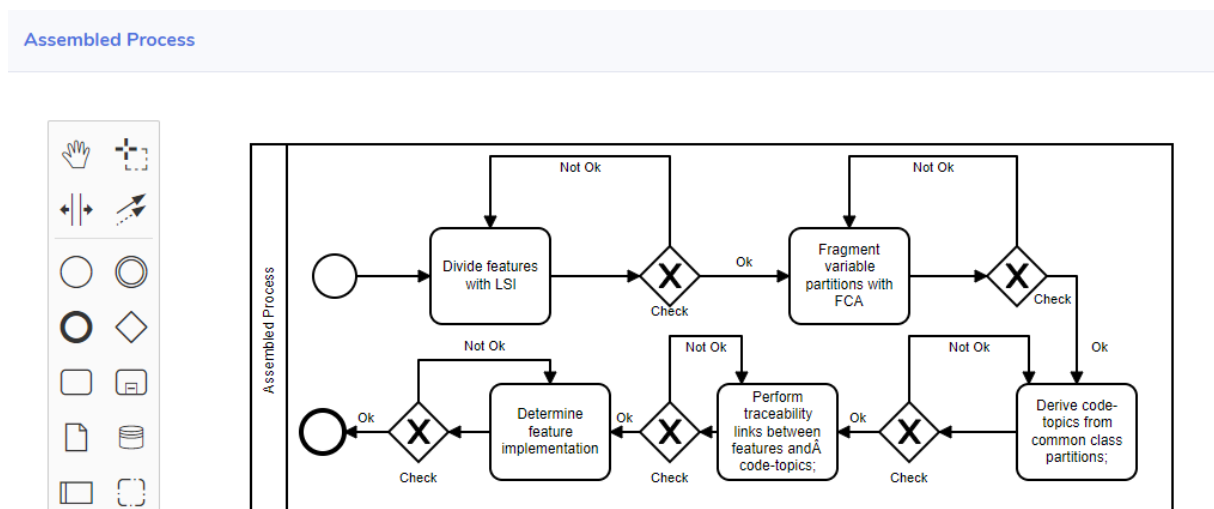
Create Feature Model:

New Activity +

Name	Order	Retrieval Tech.	Action
Perform traceability links between features and code-topics;	1	Latent Semantic Indexing	View Edit
Determine feature implementation	2	Rule Based	View Edit

Source: Author

Figure 25 – BPMN Modeling Screen



Source: Author

alternative, or OR-alternative. The user may also add artifacts related to each feature. The tool allows the user to download a feature report containing all details of the features and their related artifacts. An example is presented in Figure 28 where the features text editing system, file management, and basic are shown. In this case, the feature file man-

Figure 26 – Execute Activities Screen

The screenshot displays the 'Activity Details' interface. At the top, there are three fields: 'Name' (Fragment variable partitions with FCA), 'Order' (2), and 'Retrieval Technique' (Formal Concept Analysis). Below these is a 'Description' field containing the text 'Fragment variable partitions into minimal disjoint sets using FCA'. There are 'Pause' and 'Conclude' buttons. The 'Input Artifacts' and 'Output Artifacts' sections each have an 'Add Artifact +' button. At the bottom, a table lists artifacts with columns for Name, Type, Status, Last Update Date, and Action.

Name	Type	Status	Last Update Date	Action
Minimal disjoint sets	Design	Created	06-26-2020	View Edit Remove

Source: Author

agement had a related artifact, the minimal disjoint sets. The user can also download an XML representation of the feature model. This XML may be imported by the FeatureIDE tool, so the features may be implemented.

Our tool also provides two additional functionalities that may help the users better handle the features. One of these functionalities is presented in Figure 29, a Traceability Matrix showing which artifacts (columns) are related to each feature (rows). Lastly, the tool also allows users to configure a product from the feature model. Figure 30 shows the configuration screen, where the features may be selected based on the rules of the feature model. The product has a name and a description. After configuring a valid product, the configuration may be downloaded as XML. This XML file can be imported in the FeatureIDE. Also, a feature report of the features and artifacts present in each product may be downloaded. Besides, it is possible to visualize a matrix (Figure 31) showing the features (rows) that are present in each product (columns).

The last part of the process is to document the process experience. Here, the user may create an experience report containing details about the activities executed, including their time, issues and decisions.

Figure 27 – Feature Model Screen

Feature Model: Text Editing System

New Feature + Download Features Report FeatureIDE XML

Artifacts Traceability Matrix

Legend: Abstract; ● Mandatory; ○ Optional; ► OR Alternative; ► XOR Alternative;

Feature	Action
● TextEditingSystem	View Edit
● FileManagement	View Edit
● Basic	View Edit Remove
● Edit	View Edit
● Copy	View Edit Remove
○ SelectAll	View Edit Remove
○ Search	View Edit Remove

Source: Author

5.5 Chapter Lessons

In this chapter, we described PAXSPL framework, including the improvements related to SPL scoping. We described our process structure including its roles/actors, artifacts, phases, and activities. We also discussed how the scoping activities may aid users when conducting the reengineering. We presented the PAXSPL guidelines documentation aiming at guiding the team while applying the process, aiding users with low experience to reduce the effort of applying retrieval techniques. Besides, we presented the analysis and design of the PAXSPL supporting tool. We discussed the requirements and DD defined during development. Finally, we presented a running example.

Figure 28 – Feature Report

Feature Report

Project: Example

Feature Model: Text Editing System

Created by: Luciano

Date: 06-26-2020

1	Feature:	Text Editing System		
	Type:	Mandatory		
	Description:	SPL core.		
	Abstract:	Yes		
2	Feature:	File Management		
	Type:	Mandatory		
	Description:	Management of files		
	Abstract:	Yes		
	Artifact:1	Name:	Minimal disjoint sets	
		Type	Design	
		Description:	Variable partitions are fragmented into minimal disjoint sets using FCA.	
		Extension:	.fca	
		Link (url):	https://github.com/argouml-tigris-org/argouml	
Last Update:		06-26-2020 by Luciano		
3	Feature:	Basic		
	Type:	Mandatory		
	Description:	Basic file manager		
	Abstract:	No		

Source: Author

Figure 29 – Traceability Matrix Screen

Feature/Artifact	Minimal disjoint sets	Common and variable partitions	Code-topics	Objected-oriented Source Code	Decomposed code-topics	Traceability links
Text Editing System						
File Management	✓					
Basic						
Edit						✓
Copy				✓		
Select All	✓					

Source: Author

Figure 30 – Feature Configurator Screen

New Product

Name:
Product X2

Description:
Description

This configuration is valid and complete.

- TextEditingSystem
 - FileManagement
 - Basic
 - Edit
 - Copy
 - SelectAll
 - Search
 - Replacement
 - Help
 - ChangeDisplaySettings2
 - Resize
 - FontColor
 - Black
 - Red
 - Caseconversion
 - UpperCase
 - Lowercase

[Save Product](#)

Source: Author

Figure 31 – Product Traceability Matrix Screen

Feature Traceability Matrix: ▼		
Features/Products	EditorV1	Editorv2
TextEditingSystem	✔	✔
FileManagement	✔	✔
Basic	✔	✔
Edit	✔	✔
Copy	✔	✔
SelectAll	✔	✔
Search	✔	✔
Replacement	✔	
Help	✔	✔
ChangeDisplaySettings2	✔	✔
Resize	✔	✔
FontColor	✔	✔
Black	✔	
Red		✔
Caseconversion	✔	✔
UpperCase	✔	
Lowercase	✔	✔

Source: Author

6 EVALUATION

In this chapter, we present an initial evaluation performed on the PAXSPL framework. Section 6.1 presents the design and the procedure for conducting the evaluation on PAXSPL framework. Then, Section 6.2 presents the execution and the results of the evaluation. Lastly, Section 6.3 presents the improvements based on the answers of the RQs.

6.1 Design

In this section, we present the design of our evaluation, containing its goal, RQs, data set, and procedure. The main goal of this evaluation is:

Goal: *To measure PAXSPL customization capabilities. Thus, we aim at customizing PAXSPL to different scenarios extracted from related works and measure how PAXSPL adapts to these scenarios.*

Hence, we defined our RQs as:

- RQ1. Is PAXSPL customizable to different scenarios?** The results may evidence that our proposal is customizable and flexible to be applied in different scenarios.
- RQ2. How does PAXSPL suit different scenarios?** We intend to measure how did PAXSPL adapted to different scenarios.
- RQ3. What challenges are observed by customizing PAXSPL?** We aim at identifying and understand which challenges were faced on adapting PAXSPL.

By answering these RQs, we aim at obtaining enough evidence for achieving our goal. In this sense, **RQ1** gives evidence about PAXSPL customization capabilities. Different from **RQ1** where we only looked if the assembled process would be executed in the original scenario, in **RQ2** we look at how this assembled process would be executed. Hence, if any problems emerged, or even if some activities from the original scenario were not performed by the assembled process. Lastly, we want to identify all challenges faced when customizing PAXSPL for these different scenarios. Therefore, **RQ3** was defined for collecting and analyzing these challenges and how they have impacted the customization process.

6.1.1 Data Set

The data set used as input for this evaluation is composed of studies where SPL reengineering was applied. More specifically, we selected studies mapped in Martinez, Assunção e Ziadi (2017), which is a collaborative catalog of case studies on SPL

reengineering called Extractive Software Product Line Adoption (ESPLA). We randomly selected a study from the catalog, and by applying three criteria in each of them we would use it or not. The study should be approved in all 3 criteria to be considered for this evaluation. The criteria are:

- i **The study applied at least one retrieval technique present in PAXSPL guidelines:** Hence, we only considered studies which used retrieval techniques also covered by PAXSPL;
- ii **The study presents a scenario different from other approaches already selected:** Therefore, we guaranteed that all selected studies presented different scenarios from one another; We considered different scenarios when the study: a) used at least one different retrieval technique; b) used at least one different input artifact; or c) had a different work-flow when applying the feature retrieval techniques.
- iii **The study evaluation protocol, data set, and results are available online:** Thus, we could use their artifacts to evaluate our results.

After applying these criteria to the works mapped in Martinez, Assunção e Ziadi (2017), we have our final set of selected studies as input for this evaluation.

6.1.2 Procedure

We executed PAXSPL in each scenario extracted from the selected studies. In this case, we defined as a scenario: the artifacts, retrieval techniques, and activities of the study. Therefore, the team information used for our evaluation will be different from the original study, as we could not extract this information from the selected studies. Hence, once we started PAXSPL execution for one scenario, we intend to observe if by executing PAXSPL life-cycle, we would be able to assemble a retrieval process which would be fully executable in that scenario. Thus, we do not intend to compare the features extracted from legacy software using PAXSPL. We limited the evaluation at looking at how PAXSPL assembled a feature retrieval process representing the original process from the study. More specifically, for each study selected, we will be performing the following steps:

1. Identify and register inputs and output artifacts;
2. Identify and register feature retrieval techniques used;
3. Identify the feature retrieval activities and their work-flow;
4. Execute PAXSPL using the artifacts, techniques, and activities identified.

After executing these steps, we plan to answer the RQs by analyzing:

- The number of scenarios for which PAXSPL was able to adapt to (**RQ1 and RQ3**);
- The number of artifacts from the original study that were used by PAXSPL (**RQ2 and RQ3**);
- The retrieval techniques that were assembled into PAXSPL generic process (**RQ2 and RQ3**);
- The activities that were assembled into PAXSPL generic process (**RQ2 and RQ3**);
- The number of challenges found when adapting the scenario using PAXSPL (**RQ3**).

6.2 Execution

We included eight studies using our criteria¹, Table 10 presents the data extracted from these studies. As shown in the column artifacts, there is a variety of different artifacts from the studies, ranging from domain to development artifacts. Also, different techniques were used in these scenarios. There is also a different combination of techniques, which defines a different scenario. We executed PAXSPL using the supporting tool described in Section 5.4. The following section presents the artifacts generated during the execution.

Table 10: Data Extracted from the Original Studies

Reference	Artifacts	Techniques	Activities
Eyal-Salman, Seriai e Dony (2013)	object-oriented source code; feature descriptions	LSI; FCA; clustering;	i) Use LSI to divide features and classes into common and variable partitions; ii) fragment variable partitions into minimal disjoint sets using FCA; iii) derive code-topics from common class partition; iv) perform traceability links between features and their code-topics; v) determine which classes implement each feature.
Acher et al. (2013)	150% architecture of the system; specification of the system plugins; system plugins dependencies; architectural FM; plugin FM; constraints mapping; enhanced architectural FM;	dependency analysis; structural similarity; clustering;	i) extraction of a raw architectural FM; ii) extraction of plugin dependencies to derive inter-feature constraints from inter-plugin constraints (plugin FM); iii) automatically reconstruction of the bidirectional mapping between the architect FM and plugin FM; iv) explore the mapping to derive enhanced architectural FM.
Al-Msie'Deen et al. (2012)	object-oriented source code; object-oriented building elements; commonalities and variations; blocks of variations; atomic blocks of variation;	LSI; FCA;	i) analyze OO source code to extract OO building elements; ii) commonalities and variations are extracted using FCA (blocks of variations); iii) blocks of variations are divided into atomic blocks and features are identified based on textual similarity using FCA and LSI.

¹ Results are available at <https://docs.google.com/spreadsheets/d/1gVvkdPxDowW_kxxvX4bWrYhvm0Uj7WSFTduDHaOI98/edit?usp=sharing>

Table 10: Continued

Reference	Artifacts	Techniques	Activities
Shatnawi, Seriai e Sahraoui (2014)	object-oriented source code; component architecture; sets of component variants; concept lattice; architecture variability;	FCA; dependency analysis; ROMANTIC approach;	i) extract component-based architecture; ii) identify component variants; iii) use FCA to analyze the commonality and variability; iv) identify architecture variability.
Alves et al. (2008)	requirement documents; requirements clusters; configurations; feature model;	VSM; clustering; LSI;	i) perform a requirements similarity determination; ii) abstract requirements clusters into a configuration; iii) merge configurations for all requirements.
Chen et al. (2005)	individual requirements; requirements relationship graph; application feature trees; domain feature tree;	clustering;	i) requirements elicitation; ii) requirements relationship graph construction; iii) requirements clustering and hierarchical structure construction; iv) merging and variability modeling.
Paškevičius et al. (2012)	java source code; java .class files; dependency graph; feature distance matrix; cluster dendrogram; feature model; Feature model defined in FDL/Prolog;	clustering; dependency analysis	i) compile Java source code using a standard Java compiler ; ii) extract feature dependencies from Java class files; iii) construct a feature distance matrix; iv) cluster features based on their dependency in a feature tree; v) convert a feature tree into a FM; vi) generate description of FM in FDL/Prolog.
Breivold, Larsson e Land (2008)	architecture description; design documents; source code; user documentation; requirements specification; architecture requirements; common core assets; variable assets; SPL architecture;	dependency analysis;	i) identify requirements on the software architecture; ii) identify commonalities and variabilities; iii) restructure architecture; iv) incorporate commonality and variability; v) evaluate software architecture quality attributes.

Source: Author

6.2.1 Results

After extracting eight scenarios using the data from the original studies, we were able to assemble and execute their processes into the PAXSPL tool. In the following, we present some of the artifacts generated (by our tool) during this execution².

The first scenario was extracted from Eyal-Salman, Seriai e Dony (2013). Figure 32 presents the report of the selected retrieval techniques. These techniques are presented in detail, showing their names, description, types of input artifacts, priority order, recommendation level based on the current scenario, and the reasons for being selected. In

² Artifacts available at <<https://github.com/HestiaProject/PAXSPL/tree/master/process/evaluation>>

this sense, as we were instantiating Eyal-Salman, Seriai e Dony (2013) process, all these techniques were selected because they were used in the original study.

Figure 33 presents the BPMN representation of the assembled retrieval process. This representation is the result of assembling the techniques selected in the generic process. The five activities and their work-flow were based on the original study, however, minor modifications were made due to some challenges faced (C). First, some activities from the original study were performed in parallel (C1). As our generic process did not support this parallelism, we have to modify the original workflow. Also, not all activities were used for executing retrieval techniques (C2). This was also a challenge because our process documentation states that every activity of the assembled process should apply a retrieval technique. The last challenge faced was related to a phase of the generic process not being assembled (C3). In this case, the group phase did not have any activity assemble into it. This was also a problem as all four phases of our generic process were mandatory by definition.

Figure 32 – Techniques Report for (EYAL-SALMAN; SERIAI; DONY, 2013)

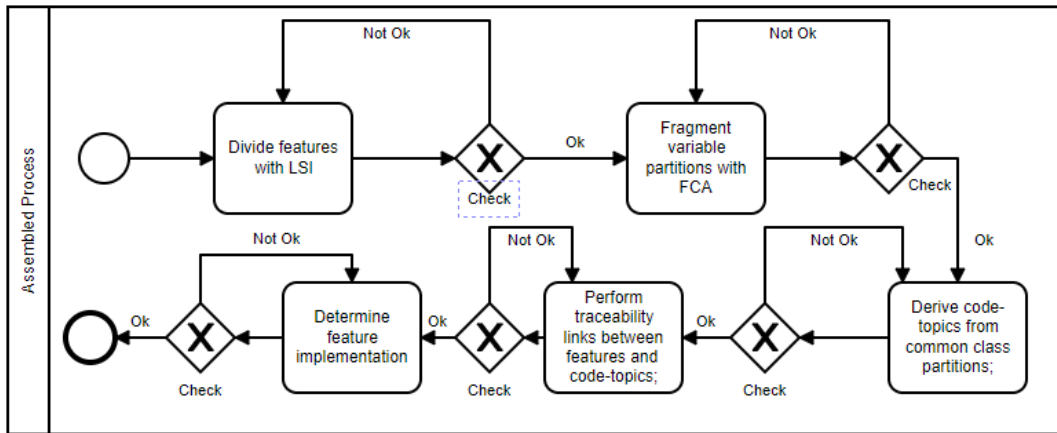
1	Name:	Latent Semantic Indexing
	Definition	A mathematical method that provides a way to identify meaningful groupings of objects that have common attributes.
	Inputs:	Development; Requirements
	Priority Order:	Extraction > Categorize > Group
	Recommendation level:	43%
	Reasons for Selection:	Chosen as it was used in the original study
2	Name:	Formal Concept Analysis
	Definition	A mathematical method that provides a way to identify meaningful groupings of objects that have common attributes.
	Inputs:	Development; Requirements; Design; Domain
	Priority Order:	Extraction > Categorize > Group
	Recommendation level:	100%
	Reasons for Selection:	It was used in the original study
3	Name:	Clustering
	Definition	Group features based on their dependencies.
	Inputs:	Development
	Priority Order:	Group > Extraction > Categorize
	Recommendation level:	43%
	Reasons for Selection:	Used to derive code-topics
4	Name:	Rule Based
	Definition	A set of rules is created to guide and help whoever is performing the feature extraction.
	Inputs:	Development; Requirements; Design; Domain
	Priority Order:	Categorize > Group > Extraction
	Recommendation level:	100%
	Reasons for Selection:	Set of rules for creating traceability links

Source: Author

Figure 34 presents part of the report detailing the assembled process. The report shows all the five activities assembled into the process as well as their input and output artifacts. In this case, the *Divide features with LSI* activity is detailed. This activity contains two input artifacts, the *Object-oriented* source code and the *Features descriptions*

and one output artifacts, the *common and variable partitions*.

Figure 33 – BPMN Process Assembled for (EYAL-SALMAN; SERIAI; DONY, 2013)



Source: Author

Figure 34 – Part of the Process Assembled Report for (EYAL-SALMAN; SERIAI; DONY, 2013)

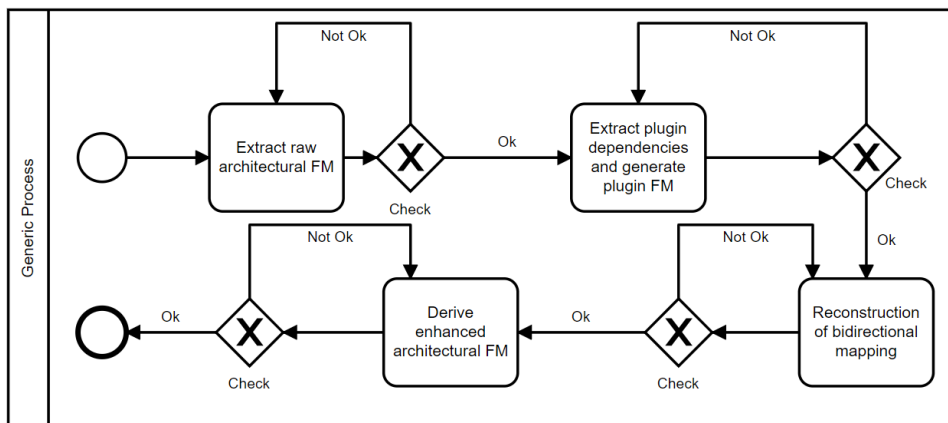
1		Phase:	Extract	
		Activity:	Divide features with LSI	
		Description:	Use LSI to divide features and classes into common and variable partitions;	
		Retrieval Technique:	Latent Semantic Indexing	
	Artifact:1	Input		
		Name:	Objected-oriented Source Code	
		Type	Development	
		Description:	Source code of the argoUML	
		Extension:	.java	
		Link (url):	https://github.com/argouml-tigris-org/argouml	
		Last Update:	05-24-2020 by Luciano	
	Artifact:2	Input		
		Name:	Feature Description	
		Type	Domain	
		Description:	Description of features of the argoUML system.	
		Extension:	.doc	
		Link (url):	https://github.com/argouml-tigris-org/argouml	
		Last Update:	05-24-2020 by Luciano	
	Artifact:3	Output		
Name:		Common and variable partitions		
Type		Development		
Description:		Classes that implement common and optional features		
Extension:		.lsi		
Link (url):		https://github.com/argouml-tigris-org/argouml		
Last Update:		05-27-2020 by Luciano		

Source: Author

The second scenario was defined by extracting the information from (ACHER et al., 2013). Figure 35 presents the BPMN representation of the assembled process. In this case, the process was composed of four activities which were described in Table 10.

Although it was possible to assemble all activities from the original study, we faced some challenges, similar to the first scenario. All three challenges faced in the first scenario were also present here, the activities not being performed in parallel (C1), an activity not using any retrieval technique (C2), and also that the group phase from the generic process was not used (C3). In addition, a new challenge was identified as one technique used in the original study was not present in PAXSPL guidelines (C4). Figure 36 presents part of the process report, showing the first activities and artifacts used as input and output for them. In this case, the plugins from the systems should be extracted to generate an architectural model, which would then be refined into a FM.

Figure 35 – BPMN Process Assembled for (ACHER et al., 2013)



Source: Author

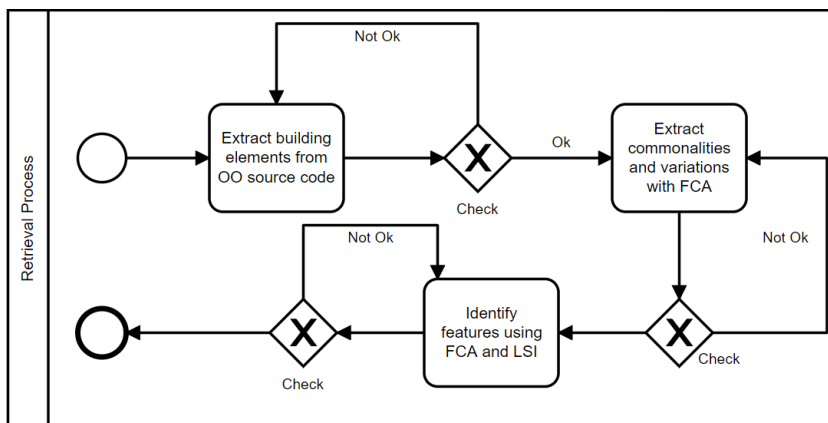
The third scenario used in the evaluation was extracted from (AL-MSIE'DEEN et al., 2012). Figure 37 presents the assembled process for the scenario, composed of three activities: extract building elements from object-oriented (OO) source code, extract commonalities and variabilities with FCA, and identify features using FCA and LSI. Considering the challenges faced when assembling this scenario, we also faced two that were presented in the previous scenarios. First, the problem with an activity not applying any retrieval technique (C2). Second, two phases (group and create FM) of the generic process were not used (C3). Also, two additional challenges were identified. The first new challenge was that one activity (**Identify features using FCA and LSI**) from the process applied more than one retrieval technique (C5). This is a challenge as in our documentation we have a relation of one retrieval technique for each activity. The second new challenge identified was that one of the activities could be part of both the extract and categorize phases of the generic process (C6). Figure 38 presents part of the process report generated in this scenario. The report describes the first two activities and their input and output artifacts. In this case, the challenge related to an activity not applying a retrieval technique is presented in the first activity where FCA is shown to be used, however, that was not the case.

Figure 36 – Process Assembled Report for (ACHER et al., 2013)

1		Phase:	Extract
		Activity:	Extract raw architectural FM
		Description:	Extract a raw architectural FM from a 150% architecture of the system
		Retrieval Technique:	Dependency Analysis
	Artifact:1	Input	
		Name:	150% architecture of the system
		Type	Architecture
		Description:	Composition of the architecture fragments of all the system plugins
		Extension:	.fm
		Link (url):	na
		Last Update:	07-23-2020 by Luciano Marchezan
	Artifact:2	Output	
		Name:	architectural FM
		Type	Architecture
Description:		Considers both the software architect viewpoint and the variability actually supported by the system.	
Extension:		.fm	
Link (url):		na	
Last Update:		07-23-2020 by Luciano Marchezan	
2		Phase:	Extract
		Activity:	Extract plugin dependencies and generate plugin FM
		Description:	Extraction of plugin dependencies to derive inter-feature constraints from inter-plugin constraints (plugin FM)

Source: Author

Figure 37 – BPMN Process Assembled for (AL-MSIE'DEEN et al., 2012)



Source: Author

We extracted the fourth scenario from (SHATNAWI; SERIAI; SAHRAOUI, 2014). In this scenario, the assembled process was composed of four activities presented in Figure 39. These four activities are: extract component-based-architecture, identify component variants, analyze commonality and variability, and identify architecture variability.

When assembling the activities, once again, the challenges related to some phases of the generic process not being used (C3) as well as a technique that was not present

Figure 38 – Process Assembled Report for (AL-MSIE'DEEN et al., 2012)

1		Phase:	Extract	
		Activity:	Extract building elements from OO source code	
		Description:	Analyze OO source code to extract OO building elements	
		Retrieval Technique:	Formal Concept Analysis	
	Artifact:1	Input		
		Name:	object oriented source code	
		Type	Development	
		Description:	Source code of the product variants	
		Extension:	.oo	
		Link (url):	na	
		Last Update:	07-23-2020 by Luciano Marchezan	
	Artifact:2	Output		
		Name:	object-oriented building elements	
		Type	Development	
Description:		packages, classes, methods, attributes of the source code		
Extension:		.doc		
Link (url):		na		
Last Update:		07-23-2020 by Luciano Marchezan		
2		Phase:	Extract	
		Activity:	Extract commonalities and variations with FCA	
		Description:	Commonalities and variations are extracted using FCA (blocks of variations)	
		Retrieval Technique:	Formal Concept Analysis	
	Artifact:1	Input		
		Name:	object-oriented building elements	
		Type	Development	
	Description:	packages, classes, methods, attributes of the source code		

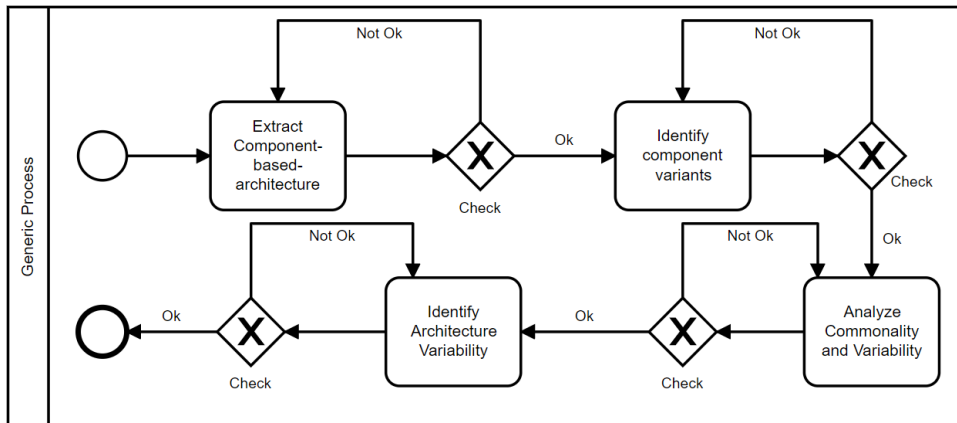
Source: Author

in PAXSPL guidelines (C4). Also, we identified a new challenge related to the final input of the approach, which was not a FM (C7). More specifically, the final activity called **Identify Architectural Variability** aimed at defining the variability of the SPL architecture using design models, such as class diagrams.

The fifth scenario used during the evaluation was extracted from (ALVES et al., 2008), and its assembled process was composed of three activities, perform a requirements similarity determination, cluster requirements to generate configuration, and merge configurations for all requirements, as illustrated by Figure 40. Only two challenges were faced during this scenario assembly, both were cited already: a phase (categorize) from the generic process was not used (C3), and that one activity (**Perform a requirements similarity determination**) applied more than one retrieval technique, characterizing C5.

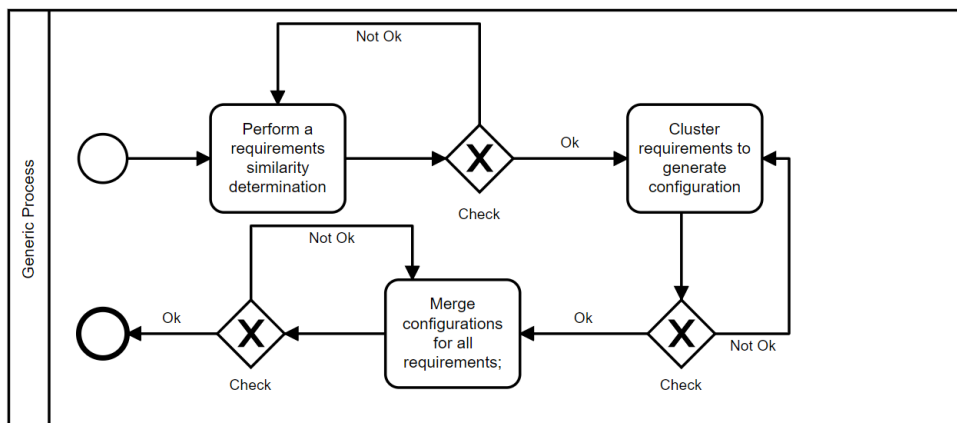
The evaluation's sixth scenario was extracted from (CHEN et al., 2005). In this scenario, the assembled process was composed of four activities as presented in Figure 41.

Figure 39 – BPMN Process Assembled for (SHATNAWI; SERIAI; SAHRAOUI, 2014)



Source: Author

Figure 40 – BPMN Process Assembled for (ALVES et al., 2008)



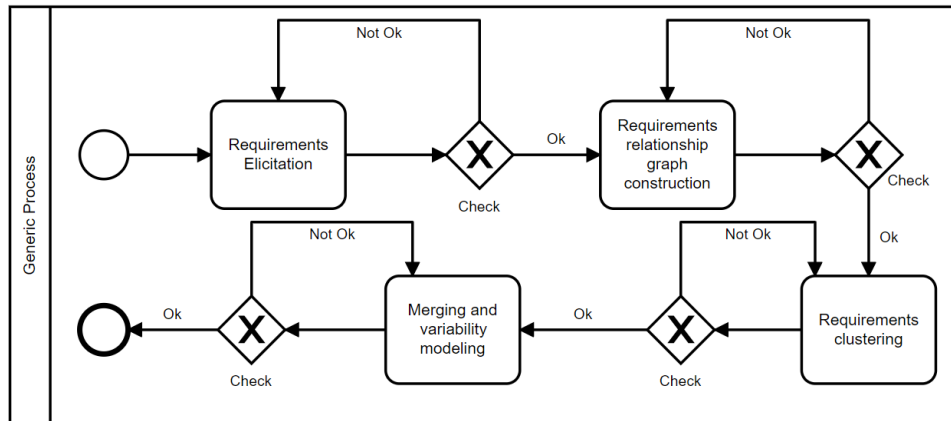
Source: Author

These activities are requirements elicitation, requirements relationship graph construction, requirements clustering, and merging variability modeling. Two challenges were faced when assembling the retrieval process. First, the problem with one activity not applying any retrieval technique reappeared (C2). Second, a new challenge was identified as the original scenario did not use input artifacts for the first activity (C8). More specifically, all artifacts from the original scenario were output from some of the activities. This may be a problem because the **Documentation Analysis** sub-process is mandatory in PAXSPL life cycle, which means that at least one artifact should be documented before the retrieval process execution.

The seventh scenario was defined by extracting the information from (PAŠKEVIČIUS et al., 2012). This scenario was composed of six activities, as illustrated in Figure 42 and only one challenge was faced. The six activities are: compile java source code, extract features dependencies from source code, construct feature distance matrix, cluster features based on dependency, convert feature tree into FM, and generate the description

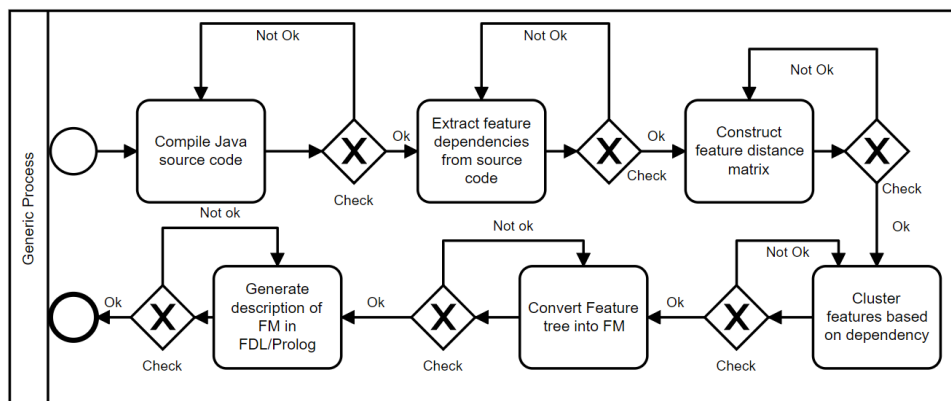
of FM in Feature Description Language (FDL) or PROLOG. The challenge was the same discussed in the previous scenario, as one activity from the assembled process did not apply any retrieval technique (C2). No new challenges were identified.

Figure 41 – BPMN Process Assembled for (CHEN et al., 2005)



Source: Author

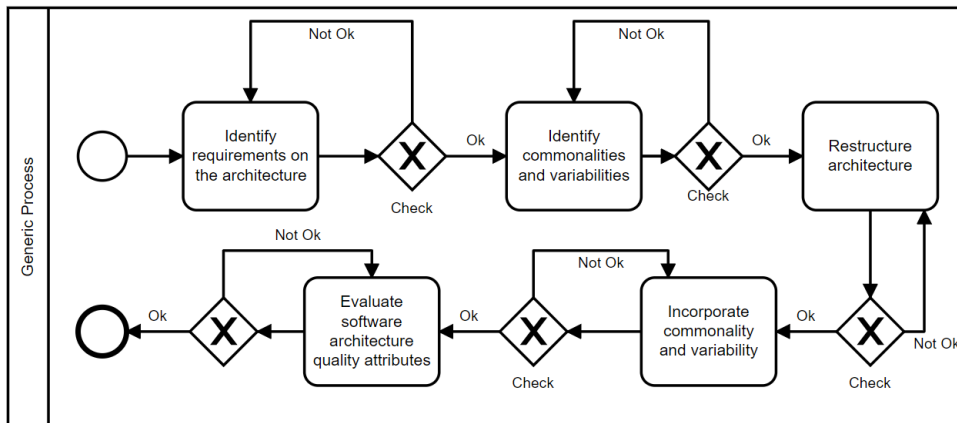
Figure 42 – BPMN Process Assembled for (PAŠKEVIČIUS et al., 2012)



Source: Author

The eighth scenario was extracted from (BREIVOLD; LARSSON; LAND, 2008). In this case, we had five activities in the assembled process. As illustrated in Figure 43, these activities are: identify requirements on the architecture, identify commonalities and variabilities, restructure the architecture, incorporate commonality and variability, and evaluate software architecture quality attributes. As happened with the previous scenario, no new challenges were identified, however, three were faced once again. First, the most recurrent one, an activity that did not apply any retrieval technique (C2). Second, phases from the generic process which were not used (C3). Lastly, the original study did not aim at generating a FM (C7).

Figure 43 – BPMN Process Assembled for (BREIVOLD; LARSSON; LAND, 2008)



Source: Author

6.2.2 Discussion

With the results of the evaluation, we were able to collect enough evidence for answering our RQs. In this section, we discuss these results.

Is PAXSPL customizable to different scenarios?

Our first questions, RQ1, aimed at measuring if our framework was customizable as planned. The protocol of our evaluation was specifically designed based on this RQ as we selected eight different scenarios from real case studies already published, which were part of the ESPLA catalog (MARTINEZ; ASSUNÇÃO; ZIADI, 2017). As presented in Table 10, all eight scenarios were different, using a variety of different artifacts, techniques, and composed of different activities. In total, when executing the scenarios using PAXSPL, more than 20 different types of artifacts were handled for our framework. Also, five different retrieval techniques were used, and their different combinations (*e.g.*, FCA and LSI) made all scenarios unique. This variety among the scenarios, along with side the fact we could customize eight assembled processes according to them, gives us evidence suggesting that our framework is indeed customizable for different scenarios. Some challenges were faced when conducting customization. These challenges were not crucial problems, however, they must be considered aiming at improving our proposal. These challenges are discussed in Section 6.2.4.

6.2.3 How does PAXSPL suit different scenarios?

As stated before, we were able to customize our framework without major problems. Table 11 summarizes these results, showing how many artifacts, activities, and techniques from the original studies were supported by PAXSPL. All artifacts used in the original studies could also be used in PAXSPL, which shows that concerning artifacts flexibility, our framework shows satisfactory results. The same results apply to the activities assembled, as all activities from the original studies could be assembled into PAXSPL

generic process. However, we faced a few challenges. Some of these challenges directly impact the results of this RQ. For instance, some techniques from the original studies were not part of PAXSPL guidelines. This means that these techniques could not be assembled into the retrieval process, as they are not presented in our documentation or tool. This challenge (C4) appeared in two scenarios, (ACHER et al., 2013) and (SHATNAWI; SERIAI; SAHRAOUI, 2014).

The more impactful challenges, however, were those related to the activities. Although all activities could be assembled for each scenario, minor modifications were performed. The first modification was related to some scenarios having parallel activities, which was not supported in our framework. The second problem, the most frequent, happened when one activity from the original study did not apply a retrieval technique. Our documentation determines that all activities in the assembled process should apply a retrieval technique. A related problem occurred when more than one retrieval technique was used in the same activity from the original study. Although these challenges impacted the PAXSPL customizability, our framework still provided satisfactory results in terms of all three variables analyzed: artifacts, techniques, and activities.

Table 11 – Results from the Evaluation.

Reference	Art.		Tech.		Act.		Challenges							
	O	P	O	P	O	P	C1	C2	C3	C4	C5	C6	C7	C8
Eyal-Salman, Seriai e Dony (2013)	6	6	4	4	5	5	✓	✓	✓					
Acher et al. (2013)	7	7	3	2	4	4	✓	✓	✓	✓				
Al-Msie'Deen et al. (2012)	5	5	3	3	3	3		✓	✓		✓	✓		
Shatnawi, Seriai e Sahraoui (2014)	5	5	3	2	4	4			✓	✓				✓
Alves et al. (2008)	4	4	3	2	3	3			✓		✓			
Chen et al. (2005)	4	4	1	1	4	4		✓						✓
Paškevičius et al. (2012)	7	7	2	2	6	6		✓						
Breivold, Larsson e Land (2008)	9	9	1	1	5	5		✓	✓					✓

O - Original Study; P - PAXSPL Support.

Source: Author.

6.2.4 What challenges are observed by customizing PAXSPL?

As discussed, eight different challenges were identified during the evaluation. Table 12 summarizes these challenges identified in this evaluation and also present possible solutions to address them. We have also to mention that these challenges affect our framework as a whole (process, guidelines, and tool). Thus, the solution may also require modifications in the whole framework.

The first challenge (C1) occurred in the first two scenarios. As our generic process does not support parallel activities, we had to handle this problem by transforming these activities into linear ones. However, a Possible Solution (PS) (PS1), would be to change the generic process and its documentation to support parallel activities.

The second challenge (C2) was the most frequent, appearing in six scenarios during the evaluation. This challenge occurred when an activity from the original study did not apply any retrieval technique. After analyzing this problem, we argue that PS2 may address it by changing our documentation to allow the users to assemble activities in the generic process without the need of assembling retrieval techniques to them.

Challenge C3 was related to some phases of the generic process not receiving activities during the assembly. For instance, in the second scenario, the group phase was not used. This was not a major problem as the generic process can be modified. As defined in its BPMN representation, however, all its phases are mandatory. We plan to solve this problem by changing the generic process to make its activities Or-optional, which means that at least one is mandatory (PS3).

Challenge C4 was more related to the guidelines than the process itself. This challenge was faced twice when techniques that were not present in the guidelines were used. This does not stop the user from applying the retrieval process as they can add the technique manually in their project. However, as we desire PAXSPL to reduce the complexity of conducting the retrieval process and also to evolve with the help of the community, we believe that C4 can be addressed by allowing the users to add or suggest the addition of new retrieval techniques into our guidelines (PS4). This could be done by providing a template or online form for users to fill and send us.

Challenges C5 and C6 are somehow similar as they both were faced due to PAXSPL current lack of handling multiple facets at the same time. First, C5 was identified when one of the activities from the original study applied more than one retrieval techniques at the same time, complementing each other. As our documentation does not give enough information about how this situation should be handled, we should include it on it PS5. The next challenge, C6, is related to one activity being part of more than one phase of the generic process, for instance, extract and categorize. The possible solution, PS6, would be similar as we should clarify in the documentation that this can be done.

The seventh challenge identified (C7) is not trivial to analyze. It was faced when two of the scenarios did not aim at generating a FM at the end of their process. This challenge is different because PAXSPL final output should be the FM as defined in our framework. However, we still understand this as an opportunity for improvement, as a company works or desires to have a variability artifact different from a FM, such as a SPL architecture. Thus, we plan to change the last activity of the generic process, which is called **Create Feature Model** to “create variability model” PS7. We also plan to clarify this in the documentation, as a variability model may be a FM, SPL architecture, or something different.

The last challenge (C8) impacts the artifacts used. This challenge appeared only in one scenario (CHEN et al., 2005), where their approach did not have input only artifact. We consider an input only artifact to be some artifact that already existed

before the execution of the retrieval process. Input only artifacts are those collected and analyzed during the **Perform Documentation Analysis** sub-process. To address this, we would change the whole documentation analysis sub-process to be optional (PS8), as for in the current version at least one activity for this process should be performed.

Table 12 – Challenges Identified During the Evaluation.

ID	Description	ID	Possible Solution
C1	Some activities from the original study were performed in parallel	PS1	Change the generic process to support parallelism
C2	An activity did not apply any retrieval technique	PS2	Allow the users to define activities without the application of retrieval techniques
C3	Some phases of the generic process did not have an activity related to it	PS3	Change the generic process to make some phases optional
C4	A technique that was not present in PAXSPL guidelines was used in an activity	PS4	Allow the users to add new techniques in our guidelines by providing a form or template
C5	One activity used more than one retrieval technique	PS5	Allow the user to assemble multiple retrieval techniques into an activity
C6	One activity was part of the more than one phase of the generic process	PS6	Allow the user to create activities that are part of more than one phase
C7	The approach did not aim at generating a FM	PS7	Consider to change the last activity of the generic process
C8	The original scenario had no input artifacts	PS8	Make the whole documentation analysis sub-process optional

Source: Author.

6.3 Improvements

In this section, we present the modifications³ performed into our framework to address the PS described earlier.

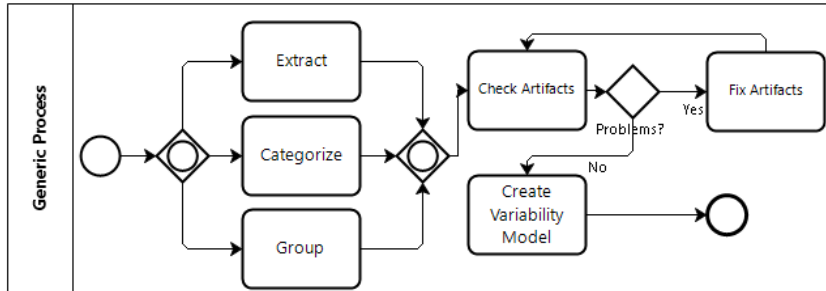
6.3.1 Modifications in the Generic Process

Three challenges (C1, C3, C7) could be mitigated by changing the generic process. Firstly, we included a parallel gateway among the extract, categorize, and group activities as illustrated by Figure 44. Also, we used an inclusion gateway, meaning that only the execution of one activity would be mandatory. Lastly, we changed the name of the last activity to **Create Variability Model**, making it more generic. These changes are highlighted and compared to the original version in Figure 45, where at the top we have the old generic process and at the bottom, we have the new one. The *A* letter points out the changes made to address challenges C1 and C3. Challenge C7 was mitigated by the part highlighted by the letter *B*. The changes applied in *A* also led to minor changes concerning the **check artifacts** activity. These changes, marked with the letter *C*, were

³ All modifications are already implemented into the documentation available at <<https://github.com/HestiaProject/PAXSPL/wiki>>

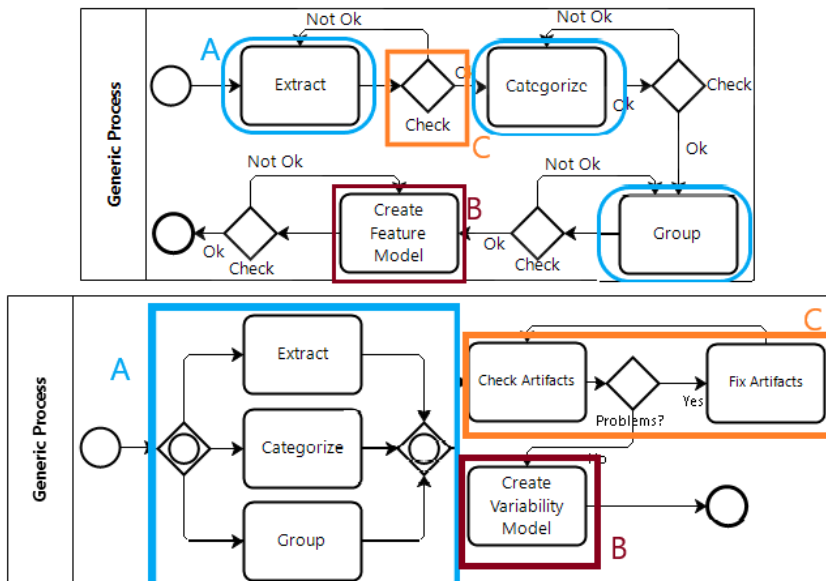
made to adapt the new parallel structure of the process. As shown, the **check artifacts** activity is executed the parallel gateway has finished, then if problems were found the artifacts should be fixed. When no problems are found, the **create variability model** activity is performed.

Figure 44 – New Generic Process for Feature Retrieval and Analysis.



Source: Author.

Figure 45 – Changes Applied to the Generic Process for Feature Retrieval and Analysis.



Source: Author.

6.3.2 Modifications in the Guidelines and Documentation

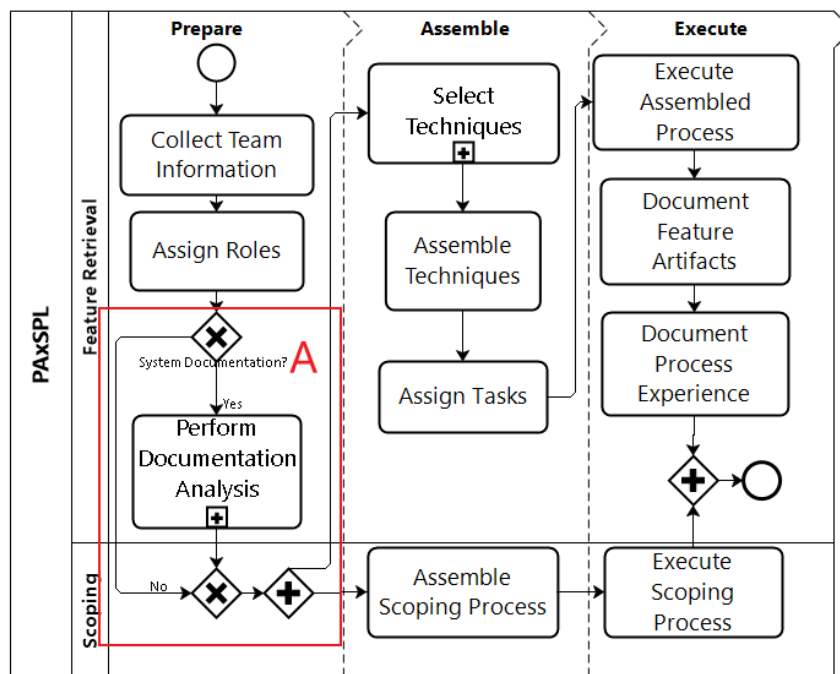
To address C2 and C5, we changed our process documentation when describing the **Assemble Techniques** activity. The new versions state *“For each activity added to the generic process, the user may choose to assemble one or more retrieval techniques from our guidelines into it”*. The first change is related to C2, where we changed the original *“should assemble”* into *“may choose to assemble”*. The second change, which is related to C5 is on *“one or more retrieval techniques”*, whereas the original version

was “a retrieval technique”. Although these are seen as minor changes, they are required to provide to users the flexibility desired from our framework.⁴ We also modified the documentation of the **Assemble Techniques** to clarify that “The activities added to the generic process may be used for different phases of the retrieval process (extract, categorize and group) depending on the user scenario”. This clarification should be enough to avoid the occurrence of C6. For addressing C4, we created an online form for users to submit new techniques into our guidelines⁵. Thus, we allowed users to contribute to the evolution of our framework as well as mitigating the occurrence of C6.

6.3.3 Modification in the PAXSPL Process

To address C8, we changed PAXSPL life-cycle. As highlighted by the letter *A* in Figure 46, we included an exclusive gateway before the documentation analysis sub-process. This sub-process will only be executed if the system’s documentation is available for the user, otherwise, this sub-process will be ignored. This addresses the challenge in scenarios where no input-only artifacts exist, such as the scenario from Chen et al. (2005).

Figure 46 – Changes Applied to PAXSPL Process.



Source: Author.

⁴ It is important to mention that these changes were still not performed in the PAXSPL tool as they required some database re-structure.

⁵ Form is available at <<https://forms.gle/Br2BjR56QhEpm2g4A>>

6.4 Chapter Lessons

In this chapter, we presented and discussed an evaluation aiming at measuring how PAXSPL is customized for different scenarios extracted from the literature. The protocol defined was discussed, as well as the reasons for selecting its RQs. We also presented the execution of its results. The results helped us to answer our RQs, showing that PAXSPL is customizable for different scenarios (RQ1) and supports almost all artifacts, activities, and techniques from the scenarios (RQ2). Some challenges were identified by answering RQ and by addressing these challenges we believe that we were able to improve our framework.

7 RELATED WORK

In this chapter, we discuss the works related to our project. We also analyze these studies and compare them with PAXSPL in terms of: SPL Reengineering in Section 7.1 and SPL Scoping in Section 7.2.

7.1 SPL Reengineering

First, we intend to perform a more detailed comparison with PAXSPL, analyzing studies that we consider more similar to ours regarding two topics of SPL reengineering: artifacts analyzed to extract features and strategies used for feature retrieval.

7.1.1 Studies Main Contribution

Considering the main contributions of the selected proposals, Acher et al. (2013) present a process, alongside a language and a tool to extract the variability from software products. They used products specification and descriptions to extract and synthesize a feature model. These product artifacts may be of different types, however, the technique used is always the merging of product descriptions using structural similarity. Despite having a similar goal to the PAXSPL, the strategy adopted by having a similar goal, Acher et al. (2013) strategy is not flexible as the user is required to perform their strategy without considering the current scenario. Bécan et al. (2013) propose a generic and ontological-aware procedure that guides users to identify siblings and parent candidates for a feature. They created and evaluated a series of heuristics for clustering and syntactic and semantic relationships, which may be considered a set of guidelines. These guidelines have a similar goal to ours, since their techniques may also be applied without prior knowledge, thus reducing the effort in comparison to other proposals.

The main contribution from Martinez et al. (2015) is a generic and extensible framework for SPL reengineering, having similar goals in comparison to PAXSPL. However, have objectives similar to ours, however, they achieved these objectives differently. On the one hand, we achieve our objectives by providing a set of guidelines to aid users when assembling a feature retrieval and analysis process to their scenario, however, our proposal cannot be extended. On the other hand, Martinez et al. (2015) contribute with a technology-based framework which does not provide the same amount of artifacts or techniques as PAXSPL, however, their framework may be expanded by extending the technologies used. In addition, their proposal intends to reduce the high investment required to adopt the SPLE. The proposed framework may be easily adapted to different artifacts types, similar to PAXSPL, and integrates state-of-the-art algorithms and visualization paradigms. Their work also presents a realization of the framework, called Bottom-Up Technologies for Reuse (BUT4Reuse).

Santos et al. (2013) propose to use testing to support the feature retrieval process. They use testing as the main input for feature extraction, and other artifacts (*e.g.*, use cases) to complement the feature retrieval process. Heuristics are used on requirements or design models, the features are mapped and test artifacts are used to expand the quality of the features retrieved. Acher et al. (2012) proposed a retrieval process focusing on plugin-based systems. Their process automatically extracts architectural feature models by combining several sources of information. These sources include software architecture, plugin dependencies, and software architectural knowledge. Their proposal also aims to give the architect the freedom to remove or create new features based on their knowledge.

Lastly, Fischer et al. (2014) study propose a novel approach for supporting clone-and-own by software engineers, the Extraction and Composition for Clone-and-Own (ECCO). Their approach is iterative, automating the Extraction and Composition steps of the clone-and-own. ECCO also provides hints guiding their users during the Completion step. This guidance can be compared to PAXSPL guidelines, however, ECCO's guidance is focused on a specific phase of their approach. Some benefits from using ECCO as described by the authors are: no need to manually find implementations of all features and features interactions, and no need to merge different implementations, preserving structure and semantics. Although ECCO and PAXSPL have some similarities such as their main goal and guidance provided, ECCO is designed specifically to be used with the clone-and-own practice, while PAXSPL aims for flexibility to the user's scenario.

7.1.2 Artifacts and Strategies for Feature Retrieval

As summarized in Table 13, these studies present some flexibility concerning artifacts and strategies used for feature retrieval. The artifacts and strategies classification used as a basis for comparison are those present in PAXSPL guidelines. However, techniques that are not present in our guidelines also appear because they were used by the related work. Considering the artifacts classification, requirements artifacts are used in (ACHER et al., 2013; BÉCAN et al., 2013; MARTINEZ et al., 2015; SANTOS et al., 2013), becoming the most common artifact amongst the approaches alongside design models which is also used in four of the studies. Domain information, however, is only used in (ACHER et al., 2012), despite being relevant in the SPL context. Architecture is also only used in (ACHER et al., 2013). Source code is used in (MARTINEZ et al., 2015; SANTOS et al., 2013; FISCHER et al., 2014). Despite these approaches using different types of artifacts, their flexibility could still be improved. For instance, in (BÉCAN et al., 2013) all artifacts are mandatory, reducing the customization according to the user's scenario. The proposal by Martinez et al. (2015), however, was designed to be generic regarding the artifact types, because gives support for integrating new artifact types in the proposal. With regard to the strategies used for feature retrieval, there is also a lack of flexibility. Despite some studies using more than one strategy type (ACHER et al.,

2013; MARTINEZ et al., 2015), as presented in Table 13, these strategies are mandatory, similar to the lack of flexibility concerning artifacts. Once again, Martinez et al. (2015) give more flexibility to the user because it allows new strategies to be integrated by extending their proposal. Also, some studies use retrieval techniques that are not present in PAXSPL guidelines. Acher et al. (2013) use structural similarity to identify architecture FM from plugins. Martinez et al. (2015) apply overlaps for identifying elements that compose an artifact, which will later be analyzed to extract features. Lastly, (ACHER et al., 2012) apply propositional logic for extracting FM from product descriptions.

Table 13 – Comparison among related works and PAXSPL (SPL Reengineering).

		Studies						
		Acher et al. (2013)	Bécan et al. (2013)	Martinez et al. (2015)	Santos et al. (2013)	Acher et al. (2012)	Fischer et al. (2014)	PAXSPL
Artifacts Type	Domain Information					✓		✓
	Requirements		✓	✓	✓			✓
	Design Models	✓		✓	✓	✓		✓
	Architecture	✓						✓
	Development			✓	✓		✓	✓
	Technological	✓						✓
Static Analysis	Clustering	✓						✓
	Dependency Analysis	✓						✓
	Data-Flow Analysis							✓
	Structural Similarity	✓						
	Overlaps			✓				
Information Retrieval	Propositional Logic					✓		
	Formal Concept Analysis							✓
	Latent Semantic Indexing							✓
Support Techniques	Vector Space Model							✓
	Expert Driven	✓		✓	✓		✓	✓
	Heuristics		✓					✓
	Rule-Based							✓

Source: (MARCHEZAN et al., 2019b).

For both topics, artifacts, and strategies, Martinez et al. (2015) give more flexibility to the users to select the most adequate option for their scenario. Still, there is a lack of flexibility for the remaining proposals. For instance, if the user's system variants possess requirements artifacts and source code only, three of the studies would not be applicable. Considering strategies, if one of the users has experience applying information retrieval strategies, they could use this experience with only one of these approaches. In addition, the studies are focused on technical aspects, which is an important point, however, there is a few concern with preparation before feature retrieval. In the related approaches, there is no activity aiming to analyze the experience and knowledge of the

user, which is a concern covered in PAXSPL. Another weak point of those studies is the lack of guidelines to help to execute their proposals since guidelines were only found in (BÉCAN et al., 2013).

7.2 SPL Scoping

We also performed a detailed comparison of a different set of related studies in terms of SPL Scoping. As these studies are detailed in Chapter 4, Section 4.2, in this section we will only focus on the direct comparisons with our work. Table 14 summarizes the SPL scoping concepts supported by all studies analyzed in our SLR with PAXSPL. As the results of the SLR were used in PAXSPL guidelines, our framework should give support to all of them. However, we should mention that this support is not broader than those presented in the works. This happens because our framework aims to guide the user when conducting the SPL scoping, however, the users are required to apply the techniques themselves. For instance, Park e Kim (2005) not only supports the metrics definition but also contains metrics that may be used by a company when scoping the SPL.

In the following section, we give an additional comparison considering the major aspects of PAXSPL.

7.2.1 Guidelines to Support SPL Scoping

Guidelines are present in Noor, Grünbacher e Hoyer (2008), CoMeS (OJEDA et al., 2018), PLEvo-Scoping (VILLELA; DÖRR; JOHN, 2010), however such guidelines are lacking information, such as the recommendation scenarios, in comparison to PAXSPL. PAXSPL guidelines aim at guiding users with low experience regarding SPL Scoping, which are not the goal of the guidelines presented by the aforementioned works. Both works use Thinklets to describe aspects of the Scoping process, such as collaborative and prioritization techniques, used for defining a product roadmap. Villela, Dörr e John (2010), on the other hand, present the use of guidelines alongside procedural descriptions, checklists, and document templates to support the SPL scoping team during the 13 activities of their process. This kind of guideline is similar to those presented in PAXSPL. In the same context, Her et al. (2007) framework contains guidelines to aid its application in different SPL projects.

7.2.2 Customization for Different Scenarios

The customization considering different scenarios and contexts is given by PuLSE-Eco (SCHMID, 2002). PuLSE-ECO is part of the PuLSE framework (DEBAUD; SCHMID, 1999), focusing on SPL scoping. As part of PuLSE, the PuLSE-Eco presents the customization of its components. Despite providing this customization, however, the PuLSE

framework does not provide some customization mechanisms as does PAXSPL, such as the generic scoping process and the feature model of scoping concepts, both presented in Chapter 4.

Table 14 – Comparison of SPL Scoping Concepts

Ref.	MD	SM	EP	PR	MA	CM	CN	PP	AD	VA	CA	FD
PuLSE			✓	✓	✓		✓		✓	✓	✓	✓
Kishi <i>et al.</i>				✓	✓			✓	✓		✓	✓
SPLSmart	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓
Park <i>et al.</i>	✓					✓						
FARE					✓	✓						
Her <i>et al.</i>	✓								✓	✓		✓
Noor <i>et al.</i>	✓			✓			✓	✓	✓	✓		
DRAMA					✓	✓	✓	✓	✓			
Planning Game in SPLE			✓				✓	✓				
CADSE		✓	✓							✓		✓
COPE+	✓		✓								✓	✓
PLEvo-Scoping	✓		✓	✓	✓					✓		✓
CAVE				✓	✓					✓		✓
PLiCs		✓					✓		✓	✓		✓
Cavalcanti <i>et al.</i>	✓	✓							✓	✓		✓
RiPLE-SC	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓
VB Portfolio Opt.		✓			✓	✓	✓				✓	✓
Acher <i>et al.</i>		✓								✓	✓	✓
Bartholdt <i>et al.</i>					✓		✓		✓	✓	✓	✓
Gillain <i>et al.</i>					✓	✓	✓					✓
Pro-PD									✓	✓		✓
ASPLE				✓					✓	✓		
Cruz <i>et al.</i>	✓					✓	✓	✓		✓	✓	✓
Nobauer <i>et al.</i>				✓	✓		✓					
Sierszecki <i>et al.</i>			✓						✓	✓		✓
SPLBench	✓						✓		✓	✓		✓
PPSMS	✓		✓		✓	✓	✓	✓		✓	✓	✓
Ianzen <i>et al.</i>										✓	✓	✓
Karimpour <i>et al.</i>	✓		✓		✓	✓	✓					✓
Neto <i>et al.</i>	✓					✓					✓	✓
ISPL			✓	✓	✓		✓		✓	✓		✓
CoMeS				✓	✓		✓	✓				✓
Small-SPL					✓		✓					
PAXSPL	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

MD - Metrics Definition; SM - Scoping Meta model; EP - Evolution Planning; PR - Product Roadmap; MA - Market Analysis; CM - Cost Modes; CN - Customer Needs; PP - Prioritize Products; AD - Architecture Definition; VA - Variability Analysis; CA - Candidates Analysis; FD - Feature Definition.

Source: Author

The work presented in Alsawalqah, Kang e Lee (2014) define a set of rules for different scenarios. Their approach handles scoping costs according to the current scenario, thus, making their approach customizable. In contrast, when defining which scoping aspects to be used in PAXSPL, the users may also handle different aspects of scoping, including costs. Other works also present some kind of customization, as we discussed in Section 4.2.1.4.

7.2.3 Approaches Domains

Lastly, the domains for which the approaches were proposed are mostly SPL in general. However, we also have a proposal for specific domains, presented in Abbas e

Andersson (2013). Their framework, called ASPLE, aims at investigating how a self-adaptive software with systematic reuse is developed as a dynamic SPL. This limits the use of the framework for other types of system, however, it is important to have works focusing on self-adaptive software, as these systems require specific development solutions (SILVA et al., 2018); Although PAXSPL is not specifically designed for self-adaptive systems development as SPL, our approach may be used for reengineering self-adaptive legacy systems, help to transform them into SPL.

RiPLE-SC (BALBINO; ALMEIDA; MEIRA, 2011) was proposed for agile methods. According to RiPLE-SC authors, SPL and agile methods share a few similar goals: to satisfy customers, increase product, decrease time-to-market, and reduce project costs. Thus, integrating SPL and agile methods, despite being challenging, may create a combined strategy to increase the benefits for both of them. Considering this context, PAXSPL's customization allows users to assemble a feature retrieval and scoping process that are suited to be executed in agile environments.

Noor, Grünbacher e Hoyer (2008) is the only approach focusing on the SPL reengineering domain, the same as PAXSPL. Thus, both approaches have similar goals, to transform legacy systems into SPL. The work of Noor, Grünbacher e Hoyer (2008), however, uses ThinkLets for describing collaborative processes through collaborative techniques. By using ThinkLets, their proposal is more robust than PAXSPL in terms of variation. However, this same robustness makes their approach less flexible in comparison to ours.

7.3 Chapter Lessons

In this chapter, we discussed works related to ours in two different aspects: SPL reengineering and SPL scoping. We analyzed these works considering different topics of interest and compare them to PAXSPL highlighting the differences and similarities among them.

8 CONCLUSION

SPL is a systematic way to reuse software assets aiming to reduce the cost of mass customization. The SPL extractive approach emerged as a solution for organizations that have a set of similar products with the potential to become SPL. SPL reengineering processes may have a huge contribution to this field, with techniques, technologies, and guidelines to help users to perform feature retrieval.

Through the analysis of a set of SPL reengineering processes mapped in Assunção et al. (2017), we established PAXSPL. PAXSPL is a process that gives support to prepare, assemble, and execute feature retrieval in a set of system variants for SPL reengineering. In previous work (MARCHEZAN et al., 2019b) we evaluated PAXSPL and defined steps for improvement: including SPL scoping specific activities into PAXSPL life-cycle, developing a supporting tool, and performing a new evaluation to measure PAXSPL customization capabilities. Thus, with the intention to investigate SPL scoping approaches, we conducted and reported an SLR in this field. With the results of this SLR, we were able to define a generic scoping process and a SPL scoping concept map, both included in the new PAXSPL version. For the next step, we defined UCs and DDs for guiding our tool development. These changes evolved PAXSPL into a framework.

To initially evaluate PAXSPL, we identified and executed eight different scenarios from the ESPLA catalog (MARTINEZ; ASSUNÇÃO; ZIADI, 2017). By assembling eight different retrieval process based on the scenarios identified, we were able to collect evidence suggesting that PAXSPL is customizable in adherence to different real scenarios. We could measure how well does PAXSPL support customization. We identified that our framework gives support to the use of several different types of artifacts, retrieval techniques, and activities. During the evaluation, however, eight challenges emerged. The identification of these challenges is important as by analyzing their impact we could propose and apply modifications into PAXSPL process, guidelines, and tools. Three modifications were applied in the generic process, which can now handle parallel and optional activities. The documentation was updated to better clarify to users how the techniques assembly can be flexible. Also, the **perform documentation** sub-process was changed to be only executed based on the user's scenario.

By concluding these steps for improvement, we believe that we achieved different contributions:

- i A SLR collecting, analyzing and discussing 33 different approaches for SPL scoping;
- ii Providing a customizable process that considers SPL scoping in different reengineering scenarios;
- iii Guidelines for users define their own customized scoping process in parallel to feature

- retrieval activities of the SPL reengineering process;
- iv A web-based, collaborative supporting tool to aid users while executing the process;
 - v An evaluation showing evidence of PAXSPL's customization capabilities in scenarios extracted from the literature;
 - vi Eight identified challenges that helped us understand PAXSPL limitations and apply improvements to address them.

As future work, we plan to conduct additional evaluations. We would like to extend our current evaluation to include additional scenarios, such as teaching classrooms scoping software evolution and software reuse. Also, we plan to conduct a case study in an organization to measure the benefits of using PAXSPL. These evaluations may also be used for collecting evidence about our tool applicability and usability. The results of these evaluations may be used to further evolve our framework. Also, several publications emerged from the results of this project. These are described as follows:

Published:

- Marchezan, Luciano, et al. "PAXSPL: A feature retrieval process for software product line reengineering." *Software: Practice and Experience* 49.8 (2019): 1278-1306. (MARCHEZAN et al., 2019b)
- Marchezan, Luciano, et al. "A Customizable SPL Scoping Process for SPL Reengineering." *Anais da III Escola Regional de Engenharia de Software*. SBC, 2019. (MARCHEZAN et al., 2019a)¹

Under Review:

- Software Product Line Scoping: A Systematic Literature Review. Submitted to *The Journal of Systems and Software*.²
- A Web-based tool for SPL reengineering. Submitted to XXXIV Brazilian Symposium on Software Engineering (Tool track).³
- PAXSPL: A feature retrieval process for software product line reengineering (Journal First Paper). Submitted to 24TH ACM International Systems and Software Product Line Conference.⁴

To be Submitted:

- Towards a Framework to Support SPL Reengineering. To be Submitted to the Third Workshop on Experiences and Empirical Studies on Software Reuse⁵

¹ Best Post-graduation Paper Award at <<http://eres.sbc.org.br/2019/>>

² JSS at <<https://ees.elsevier.com/jss/>>

³ SBES at <<http://cbsoft2020.imd.ufrn.br/sbes-ferramentas.php>>

⁴ SPLC at <<http://splc2020.net/>>

⁵ WEESR at <<https://weesr.github.io/>>

- PAXSPL: A Framework to Support Feature Retrieval and Analysis. To be Submitted to a Journal.

BIBLIOGRAPHY

- ABBAS, N.; ANDERSSON, J. Architectural reasoning for dynamic software product lines. In: **Proceedings of the 17th International Software Product Line Conference Co-located Workshops**. New York, NY, USA: ACM, 2013. (SPLC '13 Workshops), p. 117–124. ISBN 978-1-4503-2325-3. Available in: <<http://doi.acm.org/10.1145/2499777.2500718>>. Access in: 10 jun 2020. Cited 6 times at pages 44, 51, 56, 59, 61, and 114.
- ACHER, M. et al. Extraction and evolution of architectural variability models in plugin-based systems. **Software & Systems Modeling**, Springer, v. 13, n. 4, p. 1367–1394, 2013. Available in: <<https://link.springer.com/article/10.1007/s10270-013-0364-2>>. Access in: 16 april 2020. Cited 10 times at pages 11, 12, 93, 96, 97, 98, 103, 109, 110, and 111.
- ACHER, M. et al. On extracting feature models from product descriptions. In: **Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems**. New York, NY, USA: ACM, 2012. (VaMoS '12), p. 45–54. ISBN 978-1-4503-1058-1. Available in: <<http://doi.acm.org/10.1145/2110147.2110153>>. Access in: 10 jun 2020. Cited 7 times at pages 43, 50, 56, 63, 64, 110, and 111.
- AL-MSIE'DEEN, R. et al. An approach to recover feature models from object-oriented source code. **Actes de la Journée Lignes de Produits**, p. 15–26, 2012. Available in: <<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.373.1958&rep=rep1&type=pdf>>. Access in: 16 april 2020. Cited 7 times at pages 12, 22, 93, 97, 98, 99, and 103.
- ALAM, M. M.; KHAN, A. I.; ZAFAR, A. An empirical study of the improved spld framework using expert opinion technique. **IJEACS) International Journal of Engineering and Applied Computer Science**, v. 2, n. 03, 2017. Available in: <<http://ijeacs.com/Files/Other/Journal-V02-I03/An-Empirical-Study-of-the-Improved-SPLD-Framework-using-Expert-Opinion-Technique.pdf>>. Access in: 18 jun 2020. Cited 5 times at pages 44, 53, 56, 58, and 59.
- ALAM, M. M.; KHAN, A. I.; ZAFAR, A. A secure framework for software product line development. **International Journal of Computer Applications**, Foundation of Computer Science, v. 975, p. 8887, 2017. Available in: <https://www.researchgate.net/profile/Asif_Khan67/publication/313767271_A_Secure_Framework_for_Software_Product_Line_Development/links/58a55e5892851cf0e393144c/A-Secure-Framework-for-Software-Product-Line-Development.pdf>. Access in: 18 jun 2020. Cited 6 times at pages 44, 53, 56, 59, 63, and 64.
- ALSAWALQAH, H. I.; KANG, S.; LEE, J. A method to optimize the scope of a software product platform based on end-user features. **Journal of Systems and Software**, v. 98, p. 79 – 106, 2014. ISSN 0164-1212. Available in: <<http://www.sciencedirect.com/science/article/pii/S0164121214001861>>. Access in: 10 jun 2020. Cited 10 times at pages 44, 52, 54, 56, 57, 60, 61, 62, 64, and 113.
- ALVES, V. et al. Requirements engineering for software product lines: A systematic literature review. **Information and Software Technology**, Elsevier, v. 52, n. 8, p.

806–820, 2010. Available in: <<https://www.sciencedirect.com/science/article/abs/pii/S0950584910000625>>. Access in: 26 jun 2020. Cited at page 66.

ALVES, V. et al. An exploratory study of information retrieval techniques in domain analysis. In: **12th International Software Product Line Conference**. [S.l.]: IEEE, 2008. p. 67–76. Available in: <<https://ieeexplore.ieee.org/abstract/document/4626841/>>. Access in: 16 april 2020. Cited 6 times at pages 12, 74, 94, 99, 100, and 103.

AMPATZOGLOU, A. et al. Identifying, categorizing and mitigating threats to validity in software engineering secondary studies. **Information and Software Technology**, Elsevier, v. 106, p. 201–230, 2019. Available in: <<https://www.sciencedirect.com/science/article/abs/pii/S0950584918302106>>. Access in: 4 may 2020. Cited at page 65.

ASSUNÇÃO, W. et al. Reengineering legacy applications into software product lines: a systematic mapping. **Empirical Software Engineering**, Springer, p. 1–45, 2017. Available in: <<https://link.springer.com/article/10.1007/s10664-017-9499-z>>. Access in: 10 april 2020. Cited 8 times at pages 22, 25, 28, 29, 35, 66, 79, and 115.

BALBINO, M.; ALMEIDA, E.; MEIRA, S. An agile scoping process for software product lines. In: **Proceedings of SEKE**. [S.l.: s.n.], 2011. p. 717–722. Available in: <<https://www.semanticscholar.org/paper/An-Agile-Scoping-Process-for-Software-Product-Lines-Balbino-Almeida/2fa671fd0d418f3ae4bbc00a26f19d55f432733a>>. Access in: 10 jun 2020. Cited 10 times at pages 32, 43, 50, 56, 59, 60, 61, 63, 64, and 114.

BARTHOLDT, J.; BECKER, D. Scope extension of an existing product line. In: **Proceedings of the 16th International Software Product Line Conference - Volume 1**. New York, NY, USA: ACM, 2012. (SPLC '12), p. 275–282. ISBN 978-1-4503-1094-9. Available in: <<http://doi.acm.org/10.1145/2362536.2362573>>. Access in: 10 jun 2020. Cited 4 times at pages 43, 51, 56, and 59.

BAYER, J. et al. Pulse-i: Deriving instances from a product line infrastructure. In: **Proceedings Seventh IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS 2000)**. [S.l.: s.n.], 2000. p. 237–245. ISSN null. Available in: <<https://ieeexplore.ieee.org/abstract/document/839882/>>. Access in: 18 jun 2020. Cited 5 times at pages 42, 47, 55, 57, and 62.

BAYER, J.; MUTHIG, D.; WIDEN, T. Customizable domain analysis. In: **Proceedings of the First International Symposium on Generative and Component-Based Software Engineering**. London, UK, UK: Springer-Verlag, 2000. (GCSE '99), p. 178–194. ISBN 3-540-41172-0. Available in: <<http://dl.acm.org/citation.cfm?id=645416.652056/>>. Access in: 10 jun 2020. Cited 5 times at pages 32, 42, 47, 54, and 57.

BÉCAN, G. et al. Breathing ontological knowledge into feature model management. 2013. Available in: <<https://hal.inria.fr/hal-00874867/>>. Access in: 26 april 2020. Cited 5 times at pages 30, 109, 110, 111, and 112.

BOSCH, J. The challenges of broadening the scope of software product families. **Communications of the ACM**, ACM New York, NY, USA, v. 49, n. 12, p. 41–44, 2006. Available in: <<https://cacm.acm.org/magazines/2006/12/5762-the-challenges-of-broadening-the-scope-of-software-product-families/fulltext>>. Access in: 26 jun 2020. Cited at page 33.

- BOSCH, J. From software product lines to software ecosystems. In: **Proceedings of the 13th International Software Product Line Conference**. USA: Carnegie Mellon University, 2009. (SPLC '09), p. 111–119. Available in: <<https://pdfs.semanticscholar.org/7693/a20d5e97d0fe40c93ead285d2c3625a7d650.pdf>>. Access in: 18 jun 2020. Cited at page 33.
- BREIVOLD, H. P.; LARSSON, S.; LAND, R. Migrating industrial systems towards software product lines: Experiences and observations through case studies. In: **2008 34th Euromicro Conference Software Engineering and Advanced Applications**. [S.l.: s.n.], 2008. p. 232–239. Available in: <<https://ieeexplore.ieee.org/abstract/document/4725727/>>. Access in: 4 may 2020. Cited 5 times at pages 12, 94, 101, 102, and 103.
- CARBON, R. et al. Providing feedback from application to family engineering - the product line planning game at the testo ag. In: **2008 12th International Software Product Line Conference**. [S.l.: s.n.], 2008. p. 180–189. Available in: <<https://ieeexplore.ieee.org/document/4626852>>. Access in: 18 jun 2020. Cited 6 times at pages 42, 49, 55, 59, 63, and 64.
- CAVALCANTI, Y. a. C. et al. Towards metamodel support for variability and traceability in software product lines. In: **Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems**. New York, NY, USA: ACM, 2011. (VaMoS '11), p. 49–57. ISBN 978-1-4503-0570-9. Available in: <<http://doi.acm.org/10.1145/1944892.1944898>>. Access in: 10 jun 2020. Cited 6 times at pages 43, 50, 55, 57, 61, and 63.
- CHEN, K. et al. An approach to constructing feature models based on requirements clustering. In: **Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on**. [S.l.]: IEEE, 2005. p. 31–40. Available in: <<https://ieeexplore.ieee.org/abstract/document/1531025/>>. Access in: 16 april 2020. Cited 7 times at pages 12, 94, 99, 101, 103, 104, and 107.
- CHIKOFSKY, E.; CROSS, J. Reverse engineering and design recovery: A taxonomy. **IEEE software**, IEEE, v. 7, n. 1, p. 13–17, 1990. Available in: <<https://ieeexplore.ieee.org/abstract/document/43044>>. Access in: 10 jun 2020. Cited at page 28.
- CHRISTENSEN, A.; MØLLER, A.; SCHWARTZBACH, M. Precise analysis of string expressions. **Static Analysis**, Springer, p. 1076–1076, 2003. Available in: <https://link.springer.com/chapter/10.1007/3-540-44898-5_1>. Access in: 10 jun 2020. Cited at page 29.
- CLEMENTS, P.; NORTHROP, L. **Software product lines**. [S.l.]: Addison-Wesley,, 2002. Cited 2 times at pages 25 and 30.
- CRUZ, J. et al. Toward a hybrid approach to generate software product line portfolios. In: **2013 IEEE Congress on Evolutionary Computation**. [S.l.: s.n.], 2013. p. 2229–2236. ISSN 1089-778X. Available in: <<https://ieeexplore.ieee.org/document/6557834/>>. Access in: 10 jun 2020. Cited 7 times at pages 44, 51, 56, 60, 61, 63, and 64.
- CZARNECKI, K.; EISENECKER, U. W. Components and generative programming. In: SPRINGER-VERLAG. **ACM SIGSOFT Software Engineering Notes**. [S.l.],

1999. v. 24, n. 6, p. 2–19. Available in: <https://link.springer.com/chapter/10.1007/3-540-48166-4_2>. Access in: 10 jun 2020. Cited 4 times at pages 31, 32, 74, and 82.

DAVIS, S. From “future perfect”: Mass customizing. **Planning review**, MCB UP Ltd, v. 17, n. 2, p. 16–21, 1989. Available in: <<https://doi.org/10.1108/eb054249>>. Access in: 10 jun 2020. Cited at page 26.

DEBAUD, J.; SCHMID, K. A systematic approach to derive the scope of software product lines. In: **Proceedings of the 1999 International Conference on Software Engineering (IEEE Cat. No.99CB37002)**. [S.l.: s.n.], 1999. p. 34–43. ISSN 0270-5257. Available in: <<https://ieeexplore.ieee.org/abstract/document/840993/>>. Access in: 10 jun 2020. Cited 7 times at pages 42, 47, 48, 55, 57, 64, and 112.

DUMAIS, S. Latent semantic analysis. **Annual review of information science and technology**, Wiley Online Library, v. 38, n. 1, p. 188–230, 2004. Available in: <<https://asistdl.onlinelibrary.wiley.com/doi/full/10.1002/aris.1440380105>>. Access in: 10 jun 2020. Cited at page 30.

ELSNER, C. et al. Multi-level product line customization. In: **Japan Society for the Promotion of Science (JSPS); SANGIKYO Co.** Yokohama, Japan: [s.n.], 2010. Available in: <<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.391.969&rep=rep1&type=pdf>>. Access in: 10 jun 2020. Cited 4 times at pages 43, 50, 55, and 57.

ENGSTRÖM, E.; RUNESON, P. Software product line testing - a systematic mapping study. **Inf. Softw. Technol.**, Butterworth-Heinemann, USA, v. 53, n. 1, p. 2–13, jan. 2011. ISSN 0950-5849. Available in: <<https://www.sciencedirect.com/science/article/abs/pii/S0950584910001709>>. Access in: 26 jun 2020. Cited at page 66.

ESTUBLIER, J.; DIENG, I. A.; LEVEQUE, T. Software product line evolution: The selecta system. In: **Proceedings of the 2010 ICSE Workshop on Product Line Approaches in Software Engineering**. New York, NY, USA: ACM, 2010. (PLEASE '10), p. 32–39. ISBN 978-1-60558-968-8. Available in: <<https://dl.acm.org/doi/abs/10.1145/1808937.1808942>>. Access in: 10 jun 2020. Cited 6 times at pages 43, 49, 55, 62, 63, and 64.

EYAL-SALMAN, H.; SERIAI, D.; DONY, C. Feature-to-code traceability in a collection of software variants: Combining formal concept analysis and information retrieval. In: **14th International Conference on Information Reuse and Integration**. [S.l.]: IEEE, 2013. p. 209–216. Available in: <<https://ieeexplore.ieee.org/abstract/document/6642474/>>. Access in: 26 april 2020. Cited 7 times at pages 11, 82, 93, 94, 95, 96, and 103.

EYAL-SALMAN, H.; SERIAI, D.; DONY, C. Feature location in a collection of product variants: Combining information retrieval and hierarchical clustering. In: **Software Engineering and Knowledge Engineering**. [S.l.: s.n.], 2014. p. 426–430. Available in: <<https://hal-lirmm.ccsd.cnrs.fr/lirmm-01291261/>>. Access in: 16 april 2020. Cited at page 22.

FEILER, P. H.; HUMPHREY, W. S. Software process development and enactment: Concepts and definitions. In: **IEEE. Software Process, 1993. Continuous Software Process Improvement, Second International Conference on the**. [S.l.], 1993. p. 28–40. Available in: <<https://ieeexplore.ieee.org/abstract/document/236824/>>. Access in: 10 april 2020. Cited at page 32.

FISCHER, S. et al. Enhancing clone-and-own with systematic reuse for developing software variants. In: **Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on**. [S.l.]: IEEE, 2014. p. 391–400. Available in: <<https://ieeexplore.ieee.org/abstract/document/6976105/>>. Access in: 26 april 2020. Cited 2 times at pages 110 and 111.

FRAKES, W. B.; BAEZA-YATES, R. (Ed.). **Information Retrieval: Data Structures and Algorithms**. USA: Prentice-Hall, Inc., 1992. ISBN 0134638379. Cited at page 30.

FUGGETTA, A.; NITTO, E. D. Software process. In: ACM. **Proceedings of the on Future of Software Engineering**. [S.l.], 2014. p. 1–12. Available in: <<https://dl.acm.org/doi/abs/10.1145/2593882.2593883>>. Access in: 10 april 2020. Cited at page 32.

GILLAIN, J. et al. Product portfolio scope optimization based on features and goals. In: **SPLC '12 Proceedings of the 16th International Software Product Line Conference**. [S.l.]: ACM Digital Library; New York, 2012. v. 1, p. 161–170. ISBN 978 14 503 10949. Available in: <<https://dl.acm.org/doi/abs/10.1145/2362536.2362559>>. Access in: 10 jun 2020. Cited 7 times at pages 43, 51, 56, 57, 60, 63, and 64.

HER, J. S. et al. A framework for evaluating reusability of core asset in product line engineering. **Information and Software Technology**, v. 49, n. 7, p. 740 – 760, 2007. ISSN 0950-5849. Available in: <<http://www.sciencedirect.com/science/article/pii/S095058490600111X>>. Access in: 18 jun 2020. Cited 8 times at pages 42, 48, 55, 57, 62, 63, 64, and 112.

HUBAUX, A.; HEYMANS, P.; BENAVIDES, D. Variability modeling challenges from the trenches of an open source product line re-engineering project. In: IEEE. **12th International Software Product Line Conference**. [S.l.], 2008. p. 55–64. Available in: <<https://ieeexplore.ieee.org/abstract/document/4626840/>>. Access in: 26 jun 2020. Cited at page 33.

IANZEN, A. et al. Scoping automation in software product lines. In: **Proceedings of the 17th International Conference on Enterprise Information Systems - Volume 2**. Portugal: SCITEPRESS - Science and Technology Publications, Lda, 2015. (ICEIS 2015), p. 82–91. ISBN 978-989-758-097-0. Available in: <<http://dx.doi.org/10.5220/0005372400820091>>. Access in: 10 jun 2020. Cited 5 times at pages 44, 52, 56, 63, and 64.

JAIN, A. K.; DUBES, R. C. **Algorithms for clustering data**. [S.l.]: Prentice-Hall, Inc., 1988. Cited at page 29.

JOHN, I. Using documentation for product line scoping. **Software, IEEE**, v. 27, n. 3, p. 42–47, May 2010. Available in: <<https://ieeexplore.ieee.org/abstract/document/5416671/>>. Access in: 10 jun 2020. Cited 6 times at pages 31, 43, 49, 55, 57, and 64.

JOHN, I.; EISENBARTH, M. A decade of scoping: A survey. In: **Proceedings of the 13th International Software Product Line Conference**. [S.l.: s.n.], 2009. (SPLC '09), p. 31–40. Available in: <<https://dl.acm.org/doi/abs/10.5555/1753235.1753241>>. Access in: 10 jun 2020. Cited 2 times at pages 31 and 67.

JOHN, I. et al. A practical guide to product line scoping. In: **10th International Software Product Line Conference (SPLC'06)**. [S.l.: s.n.], 2006. p. 3–12. Available in: <<https://ieeexplore.ieee.org/abstract/document/1691572/>>. Access in: 10 jun 2020. Cited 6 times at pages 42, 47, 55, 57, 62, and 64.

KANG, K. et al. **Feature-oriented domain analysis (FODA) feasibility study**. [S.l.], 1990. Available in: <<https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=11231>>. Access in: 10 jun 2020. Cited 3 times at pages 25, 30, and 31.

KANG, K. et al. Feature-oriented re-engineering of legacy systems into product line assets—a case study. In: **International Conference on Software Product Lines**. [S.l.]: Springer, 2005. p. 45–56. Available in: <https://link.springer.com/chapter/10.1007/11554844_6>. Access in: 26 april 2020. Cited 2 times at pages 22 and 25.

KARIMPOUR, R.; RUHE, G. Evolutionary robust optimization for software product line scoping: An explorative study. **Computer Languages, Systems and Structures**, v. 47, p. 189 – 210, 2016. ISSN 1477-8424. Available in: <<http://www.sciencedirect.com/science/article/pii/S1477842416301063>>. Access in: 10 jun 2020. Cited 5 times at pages 44, 53, 56, 60, and 64.

KASTNER, C. et al. Featureide: A tool framework for feature-oriented software development. In: IEEE. **2009 IEEE 31st International Conference on Software Engineering**. [S.l.], 2009. p. 611–614. Available in: <<https://ieeexplore.ieee.org/abstract/document/5070568/>>. Access in: 4 may 2020. Cited at page 84.

KHTIRA, A.; BENLARABI, A.; ASRI, B. E. Towards a requirement-based approach to support early decisions in software product line engineering. In: **2014 Second World Conference on Complex Systems (WCCS)**. [S.l.: s.n.], 2014. p. 152–157. Available in: <<https://ieeexplore.ieee.org/abstract/document/7060993/>>. Access in: 10 jun 2020. Cited 6 times at pages 44, 52, 56, 61, 63, and 64.

KHURUM, M.; GORSCHER, T. A systematic review of domain analysis solutions for product lines. **Journal of Systems and Software**, Elsevier Science Inc., USA, v. 82, n. 12, p. 1982–2003, dez. 2009. ISSN 0164-1212. Available in: <<https://dl.acm.org/doi/abs/10.1016/j.jss.2009.06.048>>. Access in: 26 jun 2020. Cited at page 66.

KIM, J.; PARK, S.; SUGUMARAN, V. Drama: A framework for domain requirements analysis and modeling architectures in software product lines. **Journal of Systems and Software**, v. 81, n. 1, p. 37 – 55, 2008. ISSN 0164-1212. Available in: <<http://www.sciencedirect.com/science/article/pii/S016412120700088X>>. Access in: 18 jun 2020. Cited 5 times at pages 42, 49, 55, 63, and 64.

KISHI, T.; NODA, N.; KATAYAMA, T. A method for product line scoping based on a decision-making framework. In: CHASTEK, G. J. (Ed.). **Software Product Lines**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. p. 348–365. ISBN 978-3-540-45652-0. Available in: <<http://www.sciencedirect.com/science/article/pii/S016412120700088X>>. Access in: 10 jun 2020. Cited 5 times at pages 42, 48, 55, 60, and 61.

KITCHENHAM, B. et al. Systematic literature reviews in software engineering - a systematic literature review. **Information and Software Technology**, v. 51, n. 1, p. 7 – 15, 2009. ISSN 0950-5849. Available in: <<https://www.sciencedirect.com/science/article/pii/S0950584908001390>>. Access in: 10 jun 2020. Cited at page 37.

KITCHENHAM, B. et al. Systematic literature reviews in software engineering – a tertiary study. **Information and Software Technology**, v. 52, n. 8, p. 792 – 805, 2010. ISSN 0950-5849. Available in: <<http://www.sciencedirect.com/science/article/pii/S0950584910000467>>. Access in: 10 jun 2020. Cited 2 times at pages 36 and 66.

KLATT, B.; KROGMANN, K.; SEIDL, C. Program dependency analysis for consolidating customized product copies. In: **IEEE International Conference on Software Maintenance and Evolution**. [S.l.]: IEEE, 2014. p. 496–500. Available in: <<https://ieeexplore.ieee.org/abstract/document/6976125/>>. Access in: 26 april 2020. Cited at page 29.

KNAUBER, P. et al. Applying product line concepts in small and medium-sized companies. **IEEE Software**, IEEE Computer Society, Los Alamitos, CA, USA, v. 17, n. 05, p. 88–95, sep 2000. ISSN 0740-7459. Available in: <<https://ieeexplore.ieee.org/abstract/document/6156717/>>. Access in: 10 jun 2020. Cited 5 times at pages 42, 47, 55, 57, and 63.

KRUCHTEN, P. **The rational unified process: an introduction**. [S.l.]: Addison-Wesley Professional, 2004. Cited at page 32.

KRUEGER, C. Easing the transition to software mass customization. In: SPRINGER. **International Workshop on Software Product-Family Engineering**. [S.l.], 2001. p. 282–293. Available in: <https://link.springer.com/chapter/10.1007/3-540-47833-7_25>. Access in: 4 may 2020. Cited 3 times at pages 22, 28, and 64.

KRUEGER, C. W. Software reuse. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 24, n. 2, p. 131–183, jun. 1992. ISSN 0360-0300. Available in: <<https://doi.org/10.1145/130844.130856>>. Access in: 18 jun 2020. Cited at page 21.

LAGUNA, M. A.; CRESPO, Y. A systematic mapping study on software product line evolution: From legacy system reengineering to product line refactoring. **Science of Computer Programming**, Elsevier, v. 78, n. 8, p. 1010–1034, 2013. Available in: <<https://www.sciencedirect.com/science/article/pii/S0167642312000895>>. Access in: 26 jun 2020. Cited at page 66.

LEE, J.; KANG, S.; LEE, D. A comparison of software product line scoping approaches. **International Journal of Software Engineering and Knowledge Engineering**, v. 20, n. 05, p. 637–663, 2010. Available in: <<https://doi.org/10.1142/S021819401000489X>>. Access in: 10 jun 2020. Cited at page 68.

LINDEN, F. Van der; SCHMID, K.; ROMMES, E. **Software product lines in action: the best industrial practice in product line engineering**. [S.l.]: Springer Science & Business Media, 2007. Cited 4 times at pages 21, 22, 25, and 26.

LOBATO, L. L. et al. Risk management in software product lines: An industrial case study. In: **2012 International Conference on Software and System Process (ICSSP)**. [S.l.: s.n.], 2012. p. 180–189. Available in: <<https://ieeexplore.ieee.org/document/6225963/>>. Access in: 18 jun 2020. Cited 6 times at pages 43, 50, 56, 59, 61, and 63.

- MACHADO, I. D. C. et al. On strategies for testing software product lines: A systematic literature review. **Information and Software Technology**, Butterworth-Heinemann, USA, v. 56, n. 10, p. 1183–1199, out. 2014. ISSN 0950-5849. Available in: <<https://www.sciencedirect.com/science/article/abs/pii/S0950584914000834>>. Access in: 26 jun 2020. Cited at page 66.
- MÆRSK-MØLLER, H. M.; JØRGENSEN, B. N. Experiences initiating software product line engineering in small teams with pulse. **PuLSE**, v. 7, p. 9, 2010. Available in: <https://www.researchgate.net/profile/Bo_Jorgensen3/publication/253650939_Experiences_Initiating_Software_Product_Line_Engineering_in_Small_Teams_with_Pulse/links/53f325990cf256ab87b07a22.pdf>. Access in: 18 jun 2020. Cited 7 times at pages 43, 47, 55, 57, 59, 63, and 64.
- MARCHEZAN, L. et al. Thoth: A web-based tool to support systematic reviews. In: **2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)**. [S.l.: s.n.], 2019. p. 1–6. Available in: <<https://ieeexplore.ieee.org/document/8870160>>. Access in: 4 may 2020. Cited at page 37.
- MARCHEZAN, L. et al. Towards a generic process for spl re-engineering. In: **Escola Regional de Engenharia de Software (ERES) 1 (1)**. [S.l.]: SBC, 2017. p. 15–22. Cited 2 times at pages 35 and 36.
- MARCHEZAN, L. et al. A customizable spl scoping process for spl reengineering. In: **Anais da III Escola Regional de Engenharia de Software**. Porto Alegre, RS, Brasil: SBC, 2019. p. 137–146. Available in: <<https://sol.sbc.org.br/index.php/eres/article/view/8506>>. Access in: 10 april 2020. Cited at page 116.
- MARCHEZAN, L. et al. Paxspl: A feature retrieval process for software product line reengineering. **Software: Practice and Experience**, v. 49, n. 8, p. 1278–1306, 2019. Available in: <<https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.2707>>. Access in: 10 april 2020. Cited 9 times at pages 22, 23, 36, 37, 69, 79, 111, 115, and 116.
- MARIMUTHU, C.; CHANDRASEKARAN, K. Systematic studies in software product lines: A tertiary study. In: **21st International Systems and Software Product Line Conference - Volume A**. New York, NY, USA: Association for Computing Machinery, 2017. (SPLC '17), p. 143–152. ISBN 9781450352215. Available in: <<https://dl.acm.org/doi/10.1145/3106195.3106212>>. Access in: 26 jun 2020. Cited at page 66.
- MARTIN, R. C. **Agile software development: principles, patterns, and practices**. [S.l.]: Prentice Hall, 2002. Cited at page 32.
- MARTINEZ, J.; ASSUNÇÃO, W. K. G.; ZIADI, T. Espla: A catalog of extractive spl adoption case studies. In: **Proceedings of the 21st International Systems and Software Product Line Conference - Volume B**. New York, NY, USA: ACM, 2017. (SPLC '17), p. 38–41. ISBN 978-1-4503-5119-5. Available in: <<https://dl.acm.org/doi/10.1145/3109729.3109748>>. Access in: 4 may 2020. Cited 4 times at pages 91, 92, 102, and 115.
- MARTINEZ, J. et al. Bottom-up adoption of software product lines: a generic and extensible approach. In: ACM. **Proceedings of the 19th International**

Conference on Software Product Line. [S.l.], 2015. p. 101–110. Available in: <<https://dl.acm.org/doi/abs/10.1145/2791060.2791086>>. Access in: 10 jun 2020. Cited 5 times at pages 22, 25, 109, 110, and 111.

MONTAGUD, S.; ABRAHÃO, S.; INSFRAN, E. A systematic review of quality attributes and measures for software product lines. **Software Quality Journal**, Springer, v. 20, n. 3-4, p. 425–486, 2012. Available in: <<https://link.springer.com/article/10.1007/s11219-011-9146-7>>. Access in: 26 jun 2020. Cited at page 66.

MORAES, M. B. S. de; ALMEIDA, E. S. de; ROMERO, S. A systematic review on software product lines scoping. In: CITESEER. **Proceedings of 6th Experimental Software Engineering Latin American Workshop (ESELAW 2009)**. [S.l.], 2009. p. 63. Available in: <<shorturl.at/ruOW7>>. Access in: 10 jun 2020. Cited at page 67.

MU, Y.; WANG, Y.; GUO, J. Extracting software functional requirements from free text documents. In: **Information and Multimedia Technology, 2009. ICIMT'09. International Conference on**. [S.l.]: IEEE, 2009. p. 194–198. Available in: <<https://ieeexplore.ieee.org/abstract/document/5381217/>>. Access in: 26 april 2020. Cited at page 30.

MULLER, J. Value-based portfolio optimization for software product lines. In: **2011 15th International Software Product Line Conference**. [S.l.: s.n.], 2011. p. 15–24. Available in: <<https://ieeexplore.ieee.org/document/6030042>>. Access in: 18 jun 2020. Cited 6 times at pages 43, 50, 56, 60, 63, and 64.

NETO, P. A. da M. S. et al. A systematic mapping study of software product lines testing. **Information and Software Technology**, Butterworth-Heinemann, USA, v. 53, n. 5, p. 407–423, maio 2011. ISSN 0950-5849. Available in: <<https://www.sciencedirect.com/science/article/pii/S0950584910002193>>. Access in: 26 jun 2020. Cited at page 66.

NETO, P. A. S. et al. A hybrid approach to suggest software product line portfolios. **Applied Soft Computing**, v. 49, p. 1243 – 1255, 2016. ISSN 1568-4946. Available in: <<http://www.sciencedirect.com/science/article/pii/S1568494616304185>>. Access in: 10 jun 2020. Cited 8 times at pages 44, 53, 56, 57, 60, 61, 63, and 64.

NÖBAUER, M.; SEYFF, N.; GROHER, I. Similarity analysis within product line scoping: An evaluation of a semi-automatic approach. In: JARKE, M. et al. (Ed.). **Advanced Information Systems Engineering**. Cham: Springer International Publishing, 2014. p. 165–179. ISBN 978-3-319-07881-6. Available in: <https://link.springer.com/chapter/10.1007/978-3-319-07881-6_12>. Access in: 18 jun 2020. Cited 4 times at pages 22, 44, 52, and 56.

NOOR, M. A.; GRÜNBAACHER, P.; BRIGGS, R. O. A collaborative approach for product line scoping: A case study in collaboration engineering. In: **Proceedings of the 25th Conference on IASTED International Multi-Conference: Software Engineering**. Anaheim, CA, USA: ACTA Press, 2007. (SE'07), p. 216–223. Available in: <<http://dl.acm.org/citation.cfm?id=1332044.1332079>>. Access in: 10 jun 2020. Cited 7 times at pages 42, 48, 55, 59, 60, 61, and 64.

NOOR, M. A.; GRÜNBAACHER, P.; HOYER, C. A collaborative method for reuse potential assessment in reengineering-based product line adoption. In:

Balancing Agility and Formalism in Software Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 69–83. ISBN 978-3-540-85279-7. Available in: <https://link.springer.com/chapter/10.1007/978-3-540-85279-7_6>. Access in: 10 jun 2020. Cited 10 times at pages 42, 48, 55, 59, 60, 61, 63, 64, 112, and 114.

OJEDA, M. C. C. et al. A collaborative method for a tangible software product line scoping. In: **2018 ICAI Workshops (ICAIW)**. [S.l.: s.n.], 2018. p. 1–6. Available in: <<https://ieeexplore.ieee.org/document/8554999>>. Access in: 10 jun 2020. Cited 5 times at pages 44, 53, 56, 57, and 112.

OJEDA, M. C. C.; RODRIGUEZ, F. A.; COLLAZOS, C. A. Identifying collaborative aspects during software product lines scoping. In: **Proceedings of the 23rd International Systems and Software Product Line Conference - Volume B**. New York, NY, USA: ACM, 2019. (SPLC '19), p. 98–105. ISBN 978-1-4503-6668-7. Available in: <<http://doi.acm.org/10.1145/3307630.3342420>>. Access in: 18 jun 2020. Cited 6 times at pages 45, 53, 56, 59, 63, and 64.

O'LEARY, P.; RICHARDSON, I. Process reference model construction: implementing an evolutionary multi-method research approach. **IET Software**, v. 6, n. 5, p. 423–430, 2012. Available in: <<https://digital-library.theiet.org/content/journals/10.1049/iet-sen.2011.0195>>. Access in: 10 april 2020. Cited at page 35.

OTSUKA, J. et al. Small inexpensive core asset construction for large gainful product line development: developing a communication system firmware product line. In: **15th International Software Product Line Conference, Volume 2**. [S.l.]: ACM, 2011. p. 20. Available in: <<https://dl.acm.org/doi/abs/10.1145/2019136.2019159>>. Access in: 26 april 2020. Cited 2 times at pages 22 and 25.

O'LEARY, P.; ALMEIDA, E. S. de; RICHARDSON, I. The pro-pd process model for product derivation within software product lines. **Information and Software Technology**, v. 54, n. 9, p. 1014 – 1028, 2012. ISSN 0950-5849. Available in: <<http://www.sciencedirect.com/science/article/pii/S0950584912000572>>. Access in: 18 jun 2020. Cited 4 times at pages 43, 51, 56, and 60.

PARK, S. Y.; KIM, S. D. A systematic method for scoping core assets in product line engineering. In: **12th Asia-Pacific Software Engineering Conference (APSEC'05)**. [S.l.: s.n.], 2005. v. 1, p. 8 pp.–. ISSN 1530-1362. Available in: <<https://ieeexplore.ieee.org/document/1607187>>. Access in: 18 jun 2020. Cited 7 times at pages 42, 48, 55, 61, 62, 64, and 112.

PAŠKEVIČIUS, P. et al. Automatic extraction of features and generation of feature models from java programs. **Information Technology And Control**, v. 41, n. 4, p. 376–384, 2012. Available in: <<https://ieeexplore.ieee.org/document/1607187>>. Access in: 26 april 2020. Cited 5 times at pages 12, 94, 100, 101, and 103.

POHL, K.; BÖCKLE, G.; LINDEN, F. van D. **Software product line engineering: foundations, principles and techniques**. [S.l.]: Springer Science & Business Media, 2005. Cited 3 times at pages 21, 25, and 26.

PRESSMAN, R. S. **Software engineering: a practitioner's approach**. [S.l.]: Palgrave Macmillan, 2005. Cited at page 32.

RAMACHANDRAN, M.; ALLEN, P. Commonality and variability analysis in industrial practice for product line improvement. **Software Process: Improvement and Practice**, v. 10, n. 1, p. 31–40, 2005. Available in: <<https://onlinelibrary.wiley.com/doi/pdf/10.1002/spip.212>>. Access in: 18 jun 2020. Cited 4 times at pages 42, 48, 55, and 61.

RUBIN, J.; CHECHIK, M. Locating distinguishing features using diff sets. In: **27th IEEE/ACM International Conference on Automated Software Engineering**. [S.l.]: ACM, 2012. p. 242–245. Available in: <<https://ieeexplore.ieee.org/abstract/document/6494926>>. Access in: 10 april 2020. Cited at page 30.

RYSSEL, U.; PLOENNIGS, J.; KABITZSCH, K. Extraction of feature models from formal contexts. In: **15th International Software Product Line Conference**. [S.l.]: ACM, 2011. p. 4. Available in: <<https://dl.acm.org/doi/10.1145/2019136.2019141>>. Access in: 16 april 2020. Cited at page 30.

SALTON, G.; WONG, A.; YANG, C.-S. A vector space model for automatic indexing. **Communications of the ACM**, ACM, v. 18, n. 11, p. 613–620, 1975. Available in: <<https://dl.acm.org/doi/abs/10.1145/361219.361220>>. Access in: 10 jun 2020. Cited at page 30.

SANTOS, A. et al. Test-based spl extraction: an exploratory study. In: **28th Annual ACM Symposium on Applied Computing**. [S.l.]: ACM, 2013. p. 1031–1036. Available in: <<https://dl.acm.org/doi/abs/10.1145/2480362.2480559>>. Access in: 26 april 2020. Cited 2 times at pages 110 and 111.

SCHMID, K. Scoping software product lines: An analysis of an emerging technology. In: **Proceedings of the First Conference on Software Product Lines : Experience and Research Directions: Experience and Research Directions**. Norwell, MA, USA: Kluwer Academic Publishers, 2000. p. 513–532. ISBN 0-79237-940-3. Available in: <<http://dl.acm.org/citation.cfm?id=355461.357568>>. Access in: 10 jun 2020. Cited 8 times at pages 42, 47, 55, 57, 63, 64, 66, and 67.

SCHMID, K. A comprehensive product line scoping approach and its validation. In: **Proceedings of the 24th International Conference on Software Engineering**. New York, NY, USA: ACM, 2002. (ICSE '02), p. 593–603. ISBN 1-58113-472-X. Available in: <<https://dl.acm.org/doi/abs/10.1145/581339.581415>>. Access in: 10 jun 2020. Cited 6 times at pages 42, 47, 55, 57, 63, and 112.

SCHMID, K. et al. Introducing the pulse approach to an embedded system population at testo ag. In: **Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005**. [S.l.: s.n.], 2005. p. 544–552. ISSN 1558-1225. Available in: <<https://ieeexplore.ieee.org/abstract/document/1553600/>>. Access in: 18 jun 2020. Cited 4 times at pages 42, 47, 55, and 57.

SCHWABER, K.; BEEDLE, M. **Agile software development with Scrum**. [S.l.]: Prentice Hall Upper Saddle River, 2002. v. 1. Cited at page 32.

SEIDL, C.; WINKELMANN, T.; SCHAEFER, I. A software product line of feature modeling notations and cross-tree constraint languages. In: **Modellierung**. [S.l.: s.n.], 2016. p. 157–172. Available in: <<https://dl.gi.de/handle/20.500.12116/821>>. Access in: 10 jun 2020. Cited 2 times at pages 45 and 82.

- SEPÚLVEDA, S.; CRAVERO, A.; CACHERO, C. Requirements modeling languages for software product lines: A systematic literature review. **Information and Software Technology**, Elsevier, v. 69, p. 16–36, 2016. Available in: <<https://www.sciencedirect.com/science/article/pii/S0950584915001494>>. Access in: 26 jun 2020. Cited at page 66.
- SHATNAWI, A.; SERIAI, A.; SAHRAOUI, H. Recovering architectural variability of a family of product variants. In: SCHAEFER, I.; STAMELOS, I. (Ed.). **Software Reuse for Dynamic Systems in the Cloud and Beyond**. Cham: Springer International Publishing, 2014. p. 17–33. ISBN 978-3-319-14130-5. Available in: <https://link.springer.com/chapter/10.1007/978-3-319-14130-5_2>. Access in: 4 may 2020. Cited 5 times at pages 12, 94, 98, 100, and 103.
- SIERSZECKI, K. et al. Extending variability management to the next level. In: **Proceedings of the 18th International Software Product Line Conference - Volume 1**. New York, NY, USA: ACM, 2014. (SPLC '14), p. 320–329. ISBN 978-1-4503-2740-4. Available in: <<http://doi.acm.org/10.1145/2648511.2648548>>. Access in: 18 jun 2020. Cited 5 times at pages 44, 52, 56, 63, and 64.
- SILVA, I. F. da et al. Software product line scoping and requirements engineering in a small and medium-sized enterprise: An industrial case study. **Journal of Systems and Software**, v. 88, p. 189 – 206, 2014. ISSN 0164-1212. Available in: <<http://www.sciencedirect.com/science/article/pii/S0164121213002598>>. Access in: 11 jun 2020. Cited 7 times at pages 44, 50, 56, 59, 61, 63, and 64.
- SILVA, J. a. P. S. da et al. A systematic literature review of uml-based domain-specific modeling languages for self-adaptive systems. In: **Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems**. New York, NY, USA: Association for Computing Machinery, 2018. (SEAMS '18), p. 87–93. ISBN 9781450357159. Available in: <<https://doi.org/10.1145/3194133.3194136>>. Access in: 18 jun 2020. Cited at page 114.
- SOUZA, I. S. et al. Evidence of software inspection on feature specification for software product lines. **Journal of Systems and Software**, v. 86, n. 5, p. 1172 – 1190, 2013. ISSN 0164-1212. Available in: <<http://www.sciencedirect.com/science/article/pii/S0164121212003251>>. Access in: 18 jun 2020. Cited 6 times at pages 44, 51, 54, 56, 63, and 64.
- STOERMER, C.; O'BRIEN, L. Map-mining architectures for product line evaluations. In: IEEE. **Software Architecture, 2001. Proc.. Working IEEE/IFIP Conference on**. [S.l.], 2001. p. 35–44. Available in: <<https://ieeexplore.ieee.org/document/948405>>. Access in: 10 jun 2020. Cited 2 times at pages 22 and 25.
- STUMME, G. Formal concept analysis. In: **Handbook on ontologies**. [S.l.]: Springer, 2009. p. 177–199. Available in: <https://link.springer.com/chapter/10.1007/978-3-540-92673-3_8>. Access in: 10 jun 2020. Cited at page 30.
- ULLAH, M. I.; RUHE, G.; GAROUSHI, V. Decision support for moving from a single product to a product portfolio in evolving software systems. **Journal of Systems and Software**, v. 83, n. 12, p. 2496 – 2512, 2010. ISSN 0164-1212. Available in:

<<http://www.sciencedirect.com/science/article/pii/S0164121210002062>>. Access in: 18 jun 2020. Cited 6 times at pages 43, 49, 55, 57, 62, and 64.

VILLELA, K.; DÖRR, J.; JOHN, I. Evaluation of a method for proactively managing the evolving scope of a software product line. In: WIERINGA, R.; PERSSON, A. (Ed.). **Requirements Engineering: Foundation for Software Quality**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 113–127. ISBN 978-3-642-14192-8. Available in: <https://link.springer.com/chapter/10.1007/978-3-642-14192-8_13>. Access in: 10 jun 2020. Cited 7 times at pages 43, 50, 55, 57, 63, 64, and 112.

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**. New York, NY, USA: ACM, 2014. (EASE '14), p. 38:1–38:10. ISBN 978-1-4503-2476-2. Available in: <<http://doi.acm.org/10.1145/2601248.2601268>>. Access in: 18 jun 2020. Cited at page 41.

WOHLIN, C. et al. **Experimentation in software engineering**. [S.l.]: Springer Science & Business Media, 2012. v. 1. Cited 2 times at pages 38 and 65.

YU, Y. et al. Mining and recommending software features across multiple web repositories. In: **5th Asia-Pacific Symposium on Internetware**. [S.l.]: ACM, 2013. p. 9. Available in: <<https://dl.acm.org/doi/abs/10.1145/2532443.2532453>>. Access in: 26 april 2020. Cited at page 22.

ZIADI, T. et al. Feature identification from the source code of product variants. In: **16th European Conference on Software Maintenance and Reengineering**. [S.l.]: IEEE, 2012. p. 417–422. Available in: <<https://ieeexplore.ieee.org/document/6178889>>. Access in: 10 april 2020. Cited 2 times at pages 22 and 25.

ANNEX A – REPORTS FROM PAXSPL TOOL

Figure 47 – Team Information Report Generated by PAXSPL Tool

Team Information Report

Project: Text Editor SPL

Created by: Luciano

Date: 04-17-2020

1	Member:	Luciano
	Email:	lamp67@hotmail.com
	Roles in company:	Developer
	Experience working with SPL	Has a few works published in the field.
	Knowledge about retrieval techniques	Applied FCA a few years ago.
	Obs:	None to be made.
	Role during re-engineering:	Feature Retriever
2	Member:	Pedro
	Email:	pedro@gmail.com
	Roles in company:	
	Experience working with SPL	
	Knowledge about retrieval techniques	
	Obs:	
	Role during re-engineering:	Feature Retriever

Source: Author.

Figure 48 – Artifacts Information Report Generated by PAXSPL Tool

Input Artifacts Report

Project: Text Editor SPL

Created by: Luciano

Date: 04-17-2020

1	Name:	Source Code
	Type	Development
	Description:	Source Code from some JAVA class
	Extension:	.java
	Link (URL):	https://github.com/HestiaProject/PAXSPL/wiki/Artifacts-Description#artifacts-type-specification
	Last Update:	04-02-2020 by Luciano
2	Name:	Domain Glossary
	Type	Domain
	Description:	Domain Glossary containing terms of the domain
	Extension:	.pdf
	Link (URL):	https://github.com/HestiaProject/PAXSPL/wiki/Artifacts-Description#artifacts-type-specification
	Last Update:	02-19-2020 by Luciano
3	Name:	Use case specification 2
	Type	Requirements
	Description:	Use case descriptions
	Extension:	.doc
	Link (URL):	https://github.com/HestiaProject/PAXSPL/wiki/Artifacts-Description#artifacts-type-specification
	Last Update:	02-29-2020 by Luciano

Source: Author.

Figure 49 – Retrieval Techniques Report Generated by PAXSPL Tool

Retrieval Techniques Selected

Project: Text Editor SPL

Created by: Luciano

Date: 04-17-2020

1	Name:	Formal Concept Analysis
	Definition	A mathematical method that provides a way to identify meaningful groupings of objects that have common attributes.
	Inputs:	Development; Requirements; Design; Domain
	Priority Order:	Extraction > Categorize > Group
	Recommendation level:	75%
	Reasons for Selection:	Users with experience
2	Name:	Latent Semantic Indexing
	Definition	A mathematical method that provides a way to identify meaningful groupings of objects that have common attributes.
	Inputs:	Development; Requirements
	Priority Order:	Extraction > Categorize > Group
	Recommendation level:	58%
	Reasons for Selection:	Users with experience
3	Name:	Clustering
	Definition	Group features based on their dependencies.
	Inputs:	Development
	Priority Order:	Group > Extraction > Categorize
	Recommendation level:	17%
	Reasons for Selection:	Users with experience

Source: Author.

INDEX

BPMN, 76, 81, 82, 84, 95, 96, 104

DD, 80–82, 84, 87, 115

FCA, 22, 30, 33, 74, 97, 102

FM, 28, 82, 97, 99–101, 104, 111

FODA, 30, 31

LSI, 30, 74, 95, 97, 102

PAxSPL, 29, 35–37, 68, 69, 72, 73, 75–
77, 79–83, 87, 91–94, 97, 99, 100,
102–104, 107–116

PS, 103–105

RQ, 37, 91–93, 102, 103, 108

SLR, 36, 37, 68, 112, 115

SMS, 35, 79

SPL, 21–23, 25–28, 30–33, 36, 37, 45–54,
57–60, 68–70, 73, 75–77, 82, 87,
91, 99, 104, 109, 110, 112–116

SPLA, 27

SPLE, 21, 22, 25, 26, 33, 49, 51, 60, 109

UC, 79–81, 115

UML, 22

VSM, 30, 74