

UNIVERSIDADE FEDERAL DO PAMPA

Luciano Marchezan

**PAXSPL: a Feature Retrieval Process for
Software Product Line Re-engineering**

Alegrete
2018

Luciano Marchezan

PA_xSPL: a Feature Retrieval Process for Software Product Line Re-engineering

Term Paper presented in Software Engineering Graduation Course of Universidade Federal do Pampa as partial requirement for title obtention of Software Engineering Bachelor

Supervisor: Prof. PhD Elder de Macedo Rodrigues

Co-supervisor: Prof. PhD Maicon Bernardino da Silveira

Alegrete
2018

Luciano Marchezan

PAxSPL: a Feature Retrieval Process for Software Product Line Re-engineering

Term Paper presented in Software Engineering Graduation Course of Universidade Federal do Pampa as partial requirement for title obtention of Software Engineering Bachelor

Term Paper presented and approved in ...June... 27 of 2018
Examination Board:



Prof. PhD Elder de Macedo Rodrigues

Supervisor
UNIPAMPA

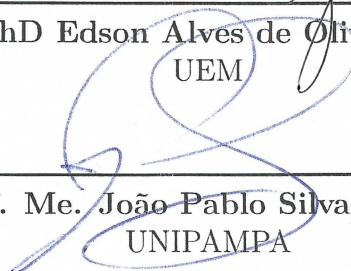


Prof. PhD Maicon Bernardino da Silveira

Co-supervisor
UNIPAMPA



Prof. PhD Edson Alves de Oliveira Junior
UEM



Prof. Me. João Pablo Silva da Silva
UNIPAMPA

“Imagination is more important than knowledge. For knowledge is limited to all we now know and understand, while imagination embraces the entire world, and all there ever will be to know and understand.” (Albert Einstein)

ABSTRACT

Software Product Lines (SPL) are a well known solution to systematically create reusable software products. Amongst the approaches to create SPL, the extractive approach is usually used when the organization already has a set of similar systems. These systems are analyzed to extract, categorize and group their common and variant features throughout the SPL re-engineering process. As there are different scenario variables, such as, available artifacts and team experience, the activities and techniques used to perform these tasks may change. This may increase the effort and decrease the precision of the retrieved features when users with low experience in SPL re-engineering perform such tasks. However, there is not a process support performing these tasks considering different scenarios.

With the objective of creating a process to be applicable in difference scenarios and flexible to fulfill the users needs, we sought in the literature and analyzed SPL re-engineering processes. We collected information about these process and compiled them to create our own. Therefore, we propose PAXSPL, a process that provides support to prepare, assemble and execute feature retrieval throughout the analysis of product artifacts, documentation and team experiences. Within PAXSPL we also included a set of guidelines to give support to those who may perform it.

To evaluate our proposal contribution to the SPL field, we applied a survey in experts of the area. The survey results gave use evidence about PAXSPL contribution and relevance for the field. To further evaluate our process we conducted and reported an exploratory case study in a real development environment. The organization where the case study was executed possesses a large number of products that are candidates to become SPL. The case study results were important to identify some points for improvement in PAXSPL. We also could use the information gathered to improve the guidelines and give these information to be use as basis of comparison to future users.

Key-words: Software Product Line Re-engineering, Software Re-engineering, Software Product Lines, Software Reuse, Software Evolution.

RESUMO

Linhas de Produto de Software (LPS) são uma solução bem conhecida para se criar de maneira sistemática softwares reutilizáveis. Entre as abordagens existentes para se criar LPS, a abordagem extrativa é geralmente utilizada quando uma empresa já tem um conjunto de sistemas similares. Esses sistemas são analisados para se extrair, categorizar e agrupar suas características em comum e variantes por meio do processo de re-engenharia de LPS. Como existem diferentes variáveis de cenário, como artefatos disponíveis e experiência da equipe, as atividades e técnicas utilizadas para se executar estas tarefas podem alterar. Isto pode aumentar o esforço e diminuir a precisão das características recuperadas, principalmente quando usuários com pouca experiência em re-engenharia de LPS executam estas tarefas. Porém, não existe um processo que de suporte para executar tais tarefas levando em consideração diferentes cenários.

Com o objetivo de se criar um processo que possa ser aplicável em diferentes cenários e flexível para atender as necessidades dos usuários, nós buscamos na literatura e analisamos processos de re-engenharia para LPS. Foram coletados dados sobre estes processos, sendo que os mesmos foram mesclados para criar nosso próprio processo. Portanto, neste trabalho propõe-se PAXSPL, um processo que fornece suporte para preparar, montar e executar a recuperação de características por meio da análise de artefatos de produto, documentação e experiência do time. Foi incluído no PAXSPL, um conjunto de diretrizes para dar suporte àqueles que venham a executá-lo.

Para avaliar a contribuição da nossa proposta para a área de LPS, aplicou-se um questionário em especialistas da área. Os resultados do questionário forneceram evidência sobre a contribuição e relevância do PAXSPL para a área. Para obter uma avaliação adicional, um estudo de caso exploratório em um ambiente real de desenvolvimento foi conduzido e reportado. A empresa onde o estudo de caso foi aplicado possui um grande número de produtos que são candidatos à se tornarem LPS. Os resultados do estudo de caso foram importantes para se identificar alguns pontos para se melhorar PAXSPL. Adicionalmente, pode-se usar as informações coletadas para incrementar as diretrizes de suporte e fornecer algumas informações coletadas para serem usadas como base de comparação para futuros usuários do processo.

Palavras-chave: Re-engenharia de Linhas de Produto de Software, Re-engenharia de Software, Linhas de Produto de Software, Reuso de Software, Evolução de Software.

LIST OF FIGURES

Figure 1 – The two-life-cycle model of SPLE.	26
Figure 2 – SPL Re-engineering process.	29
Figure 3 – A feature model of a car.	31
Figure 4 – A feature model created using the Generative Programming notation.	31
Figure 5 – The generic process for feature retrieval.	46
Figure 6 – An example of assembled process.	47
Figure 7 – The PAXSPL process.	47
Figure 8 – Documentation analysis sub-process.	49
Figure 9 – Select techniques sub-process.	53
Figure 10 – A feature model of retrieval techniques.	55
Figure 11 – Participants technical knowledge.	66
Figure 12 – Frequency diagram of the quantitative questions.	67
Figure 13 – First version of our process.	70
Figure 14 – First version of our process (Feature Search Activity).	70
Figure 15 – Team information report generated during case study.	78
Figure 16 – Retrieval techniques report generated during case study.	79
Figure 17 – Concept lattice of features extracted from ten different products of ICode.	80
Figure 18 – ICode feature model from Phase 1.	81
Figure 19 – ICode feature model from Phase 2.	82
Figure 20 – Effort percentage of each PAXSPL activity.	93
Figure 21 – Effort percentage of each PAXSPL phase.	93
Figure 22 – Template for Team Information Report	109
Figure 23 – Template for Artifacts Type Specification	111
Figure 24 – Template for Retrieval Techniques Report	113
Figure 25 – Template for Feature Report	115
Figure 26 – Template for Process Execution Report	117

LIST OF TABLES

Table 1 – Base Search String.	35
Table 2 – Reduced search string.	35
Table 3 – Retrieved studies by database.	37
Table 4 – Summary of the selection process.	38
Table 5 – Quality assessment result.	38
Table 6 – Artifacts used for each study.	39
Table 7 – Retrieval strategies used for each study.	39
Table 8 – Experience applying/performing/monitoring:	66
Table 9 – Time spent in each activity of PAXSPL during both phases of case study.	76
Table 10 – Quality of PAXSPL artifacts generated during both phases of case study.	77
Table 11 – Artifacts reused during Phase 2.	83
Table 12 – Use of artifacts for each activity of PAXSPL.	84
Table 13 – Relevance of artifacts according to survey results.	85
Table 14 – Complexity level of each PAXSPL activity according to survey answers.	86
Table 15 – Reusability level of PAXSPL generated artifacts according to survey 3 results.	86
Table 16 – Relevance score of each artifact.	88
Table 17 – Relevance level of each artifact during PAXSPL execution.	89
Table 18 – Impact level of artifacts when selecting techniques.	90
Table 19 – Effort score of each PAXSPL activity.	92
Table 20 – Reusability score of PAXSPL artifacts.	95
Table 21 – Reusability level of PAXSPL artifacts.	95

LIST OF ABBREVIATIONS

EC Exclusion Criteria

FCA Formal Concept Analysis

FODA Feature-Oriented Domain Analysis

IC Inclusion Criteria

LSI Latent Semantic Indexing

PAxSPL Prepare Assemble and Execute Process for Software Product Line Re-engineering

QA Quality/Assessment Criteria

QM Quality Metrics

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

VSM Vector Space Model

TABLE OF CONTENTS

1	INTRODUCTION	21
1.1	Motivation	21
1.2	Objectives	22
1.3	Methodology	22
1.4	Organization	23
2	BACKGROUND	25
2.1	Software Product Line	25
2.1.1	Software Product Line Engineering	25
2.1.1.1	Domain Engineering	26
2.1.1.2	Application Engineering	27
2.1.2	Software Product Line Re-engineering	27
2.1.2.1	Feature Retrieval Techniques	28
2.1.3	Feature Model	30
2.2	Software Process	31
2.3	Chapter Lessons	32
3	RELATED WORK	33
3.1	A Systematic Literature Review	33
3.1.1	Planning Review	33
3.1.1.1	Scope and Objective	33
3.1.1.2	Question Structure	33
3.1.1.3	Research Questions	34
3.1.1.4	Search Process	34
3.1.1.5	Inclusion and Exclusion Criteria	34
3.1.1.6	Quality Assessment Criteria	35
3.1.1.7	Selection Process	36
3.1.2	SLR Conduction	37
3.2	Discussing Related Work	38
3.2.1	Artifacts and Strategies for Feature Retrieval	38
3.2.2	Studies Main Contribution	39
3.2.3	Applicability in Different Scenarios	40
3.3	Chapter Lessons	41
4	THE PREPARE, ASSEMBLE AND EXECUTE PROCESS	43
4.1	Process Overview	43
4.1.1	Process Structure	43
4.1.2	Actors/Roles	44
4.1.3	Artifacts	44

4.1.3.1	Input Artifacts	45
4.1.3.2	Output Artifacts	45
4.1.3.3	Generic and Assembled Processes	46
4.2	Phases and Activities	47
4.2.1	Prepare	48
4.2.1.1	Collect Team Information	48
4.2.1.2	Assign Roles	48
4.2.1.3	Perform Documentation Analysis	48
4.2.1.3.1	Collect Domain Information	49
4.2.1.3.2	Register Domain Constraints and Vocabulary	50
4.2.1.3.3	Collect Requirements Information	50
4.2.1.3.4	Register Architectural Information	50
4.2.1.3.5	Collect Artifact Information	51
4.2.1.3.6	Register Development Information	51
4.2.1.3.7	Register Technological Information	51
4.2.2	Assemble	52
4.2.2.1	Select Techniques	52
4.2.2.2	Assemble Techniques	52
4.2.2.3	Assign Tasks	53
4.2.3	Execute	54
4.2.3.1	Execute Assembled Process	54
4.2.3.2	Document Feature Artifacts	54
4.2.3.3	Document Process Experience	54
4.3	Guidelines	55
4.3.1	Static Analysis	55
4.3.1.1	Clustering	56
4.3.1.2	Dependency Analysis	56
4.3.1.3	Data Flow Analysis	57
4.3.2	Information Retrieval	58
4.3.2.1	Formal Concept Analysis	58
4.3.2.2	Latent Semantic Indexing	59
4.3.2.3	Vector Space Model	60
4.3.3	Support Techniques	60
4.3.3.1	Expert-Driven	61
4.3.3.2	Heuristics	61
4.3.3.3	Rule-based	61
4.3.4	Support Checklist	61
4.4	Chapter Lessons	63
5	ASSESSING OUR PROPOSAL	65

5.1	Survey	65
5.1.1	Participants Profile	65
5.1.2	Results	67
5.1.3	Results Analysis	68
5.1.4	Process Improvements considering the Survey Results	69
5.2	Case Study	70
5.2.1	Rationale and Preparation	70
5.2.1.1	Objectives and Research Questions	71
5.2.1.2	Context and Units of Analysis	72
5.2.1.3	Case Study Protocol	73
5.2.1.4	Data Collection	73
5.2.1.5	Quality Metrics	74
5.2.2	Conduction	75
5.2.3	Collected Data	76
5.2.3.1	Time Spent	76
5.2.3.2	Quality of Generated Artifacts	76
5.2.3.3	Use of Generated Artifacts	83
5.2.3.4	Surveys Answers	85
5.2.4	Results and Discussion	87
5.2.4.1	RQ1. What is the impact/relevance level of the artifacts generated by PAXSPL execution in terms of re-engineering?	87
5.2.4.2	RQ2. What is the quality of the generated artifacts?	90
5.2.4.3	RQ3. What is the effort needed to perform each phase of PAXSPL?	91
5.2.4.4	RQ4. What is the reuse capability of PAXSPL?	94
5.2.5	Threats to Validity	96
5.3	Chapter Lessons	97
6	FINAL CONSIDERATIONS	99
	BIBLIOGRAPHY	101
	APPENDIX	107
	APPENDIX A – TEMPLATE FOR TEAM INFORMATION REPORT	109
	APPENDIX B – TEMPLATE FOR ARTIFACTS TYPE SPEC- IFICATION	111

**APPENDIX C – TEMPLATE FOR RETRIEVAL TECHNIQUES
REPORT 113**

APPENDIX D – TEMPLATE FOR FEATURE REPORT . 115

**APPENDIX E – TEMPLATE FOR PROCESS EXECUTION
REPORT 117**

Index 119

1 INTRODUCTION

The increasing demand for quality software and the concern of software development companies in lowering project costs has led to the emergence of software engineering solutions such as Software Product Line Engineering (SPLE) (LINDEN; SCHMID; ROMMES, 2007). As happened in the automobile industry, the introduction of product lines in software production caused big changes to the software industry. As Pohl, Böckle e Linden (2005) highlighted, the benefits and motivation for adopting Software Product Lines (SPL) are many, such as: reduced maintenance effort, reduced complexity, increased reusability and better cost estimates.

Pohl, Böckle e Linden (2005) discussed the many benefits from implementing SPL. A great example of these benefits is the cost reduction in comparison to single systems. In the case of a few systems, the costs for SPLE are relatively high, however, they are significantly lower for larger quantities.

The Software Re-engineering process is defined by Chikofsky e Cross (1990) as “the examination and alteration of a system to reconstitute it in a new form”. In the SPL context, however, this definition is more specific. It could be described as the process of evolving legacy systems into SPL by identifying and extracting common and variable points (LAGUNA; CRESPO, 2013). Whatever definition is used, the goal of the SPL Re-engineering process is to take a number of product variants and transform them into SPL with the use of techniques, methods and tools (ASSUNÇÃO et al., 2017). This goal is achieved through the analysis and extraction of features from those product variants (feature retrieval).

With the understanding of the importance of the SPL Re-engineering process to migrate from a set of system variants to SPL (KRUEGER, 2001), we decided to propose a work that intends to reduce the effort of performing this process.

With our process, we intend to provide enough flexibility with regard to artifacts, strategies and techniques used for feature retrieval. The flexibility in this case would be the capacity of the process to be easily modified according to users needs. This flexibility is given by allowing the choice and combination of different techniques for feature retrieval based on information gathered during the process execution. This information includes: team experiences and skills, domain engineering artifacts, requirements artifacts, product artifact types and extensions, and technologies used to develop the products. We created a set of guidelines to help whoever is performing our process to choose the techniques that better fit their context.

1.1 Motivation

While many systems are now being developed proactively as a SPL, there are still those that emerge when a company has one or more software products and wants to create a software product line out of them. Should they start from scratch or can they

use those software products and transform them into **SPL**? If they choose to transform their products into a product line, how can they do it?

A possible answer to the latter question is the **SPL** Re-engineering process. The extractive **SPL** approach (KRUEGER, 2001) might be used to transform such already developed software products into **SPL**. This approach uses the **SPL** re-engineering process to take a number of product variants and transform them into **SPL** with the use of techniques, methods and tools (ASSUNÇÃO et al., 2017).

To the best of our knowledge, a flexible and well defined process for performing feature retrieval in the **SPL** re-engineering process has not been proposed yet. Otsuka et al. (2011) and Ziadi et al. (2012) argue that a set of guidelines is needed to formalize this process. 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 that guidelines may lead to an automated support for this process.

The proposal of **SPL** re-engineering process would be a relevant contribution for the **SPLE** field because it may reduce the complexity and effort of **SPL** Re-engineering. In addition, a flexible process may be performed and replicated in different scenarios. This would reduce cost and effort because software engineers may use an already-performed process instead of planning it all from scratch. A scenario may be different of another by considering many points: different systems being re-engineered, different artifacts being used to extract features or even a different team working to create the **SPL**.

1.2 Objectives

The main objective of this work is to propose a process that prepares, assembles and executes feature retrieval for **SPL** re-engineering. Our process main goal is to assemble different feature retrieval processes for different scenarios, aiming for a high level reuse. The assembled process will be executed to detect and extract features from a set of product variants. This objective is divided into the following specific goals:

- Assemble a feature retrieval process for different scenario and execute it;
- Create a set of guidelines to support the process assembly. These guidelines will focus on the retrieval techniques selection;
- Perform a case study to collect exploratory evidence and evaluate our proposal in a real development environment.

1.3 Methodology

The methodology used in this work is presented in the following topics: **Analyze related work**: we conducted a systematic literature review to identify and analyze the

main contributions to the field; **Create the process:** based on the information collected from related works, our process was created and documented; **Map the retrieval techniques and create a set of guidelines:** we mapped the most used and combined techniques for feature retrieval into a set of guidelines; **Obtain an early feedback of the proposal:** a survey was applied in researchers of the area to measure the impact and contribution of our proposal; **Evaluate the proposal in a real development environment:** a case study was performed to collect exploratory evidence about our process in the industry.

1.4 Organization

This document is organized according to the following:

- **Chapter 2: Background** - Details of main concepts related to our work, such as, [SPL](#) and feature retrieval strategies;
- **Chapter 3: Related Work** - Works similar to ours ranked by quality criteria;
- **Chapter 4: Proposal** - Description of our proposed process and its guidelines;
- **Chapter 5: Assessing Our Proposal** - We describe a survey applied to researchers of the [SPLE](#) field to validate our proposals impact and relevance; In addition we report a case study performed in the industry to collect exploratory evidence about our proposal;
- **Chapter 6: Final Considerations** - We present the conclusions obtained by executing the evaluations of our proposal and planned future work as well.

2 BACKGROUND

In this chapter we review 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](#) Re-engineering. Lastly, the definition and characteristics of Feature Models are showed in Section 2.1.3.

2.1 Software Product Line

To the best of our knowledge, 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](#)).

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 e 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 for implementing [SPL](#) are: 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 the [SPL](#) development is to get a set of product variants and extract their variabilities to create the [SPL](#). This process is called Software Product Line Re-engineering ([ASSUNÇÃO et al., 2017](#)). For both process 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](#) re-engineering and feature model concepts are described in the following sections.

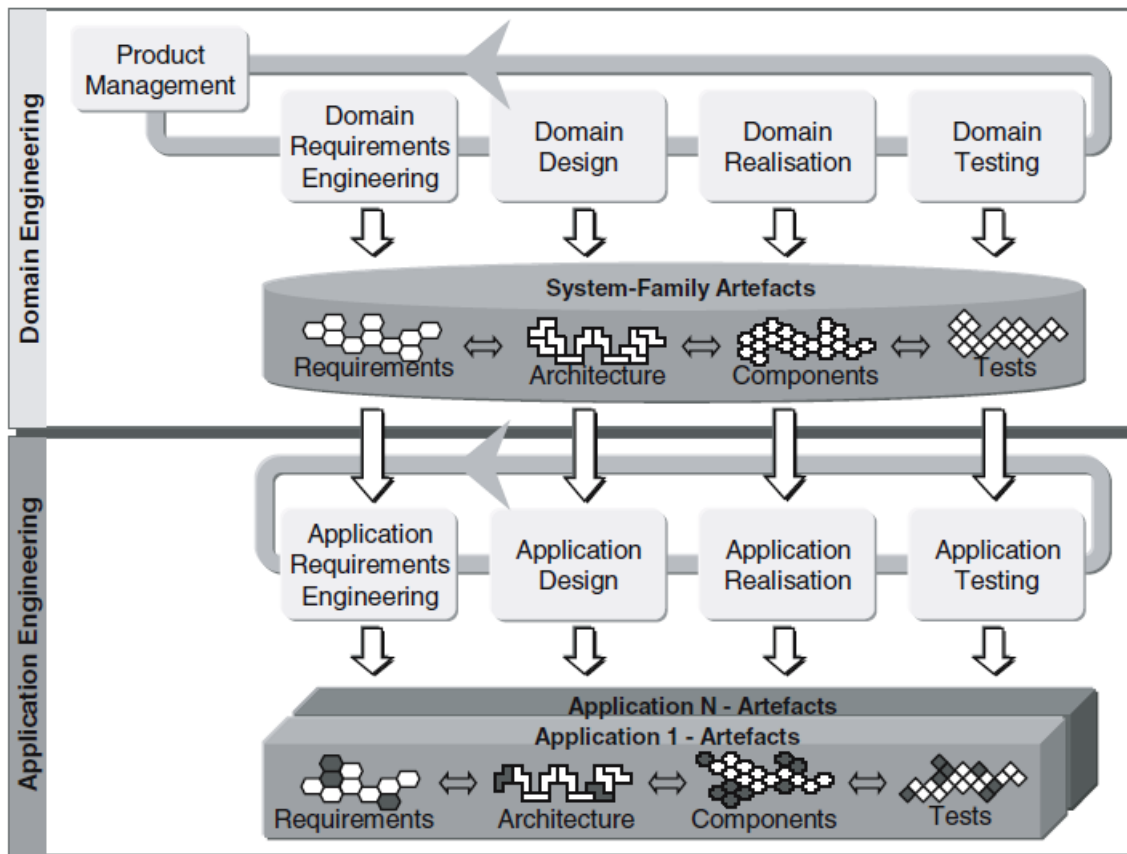
2.1.1 Software Product Line Engineering

[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: 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 e 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 focus on the definition of the SPL scope and common assets generation, from requirements to testing, that compose the SPL platform. We present a briefly description adapted from Pohl, Böckle e 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 elicitate, document, negotiate, validate and verify, and manage the requirements, a set of common variables, well-defined requirements and the [SPL](#) variability model;
- Domain Design: during this sub-process, domain requirements are used alongside the [SPL](#) variability model to define the Software Product Line Architecture ([SPLA](#));
- Domain Realisation: in this sub-process, the [SPLA](#) and other domain assets are analyzed to design and implement the software components of the [SPL](#) domain;
- Domain Testing: the domain testing sub-process is when the realisation 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 by four sub-processes are briefly described as follows:

- Application Requirements Engineering: during this sub-process 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: during the application design sub-process, 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 of this sub-process is to generate an executable software product. This generation is done by using the application architecture and the domain realisation. At the end of this sub-process 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 realisation product.

2.1.2 Software Product Line Re-engineering

According to [Krueger \(2001\)](#), there are three possible approaches for an organization moving from traditional software to SPL: proactive, reactive, and extractive.

By using the proactive approach, an organization would plan, analyze, design and develop a complete **SPL** including the whole scope of their products. With the reactive approach, the organization would increase their **SPL** development in order to reach the demand for new products, or emerging new requirements. The extractive approach, however, is used on 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 a **SPL**.

However, 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** re-engineering.

The term re-engineering 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, re-engineering 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** re-engineering process is composed of three main phases, described below:

- **Detection:** during the detection phase, variabilities and commonalities of the products are identified and extracted throughout the use of feature retrieval techniques. These variabilities and commonalities are represented in the form of features. Techniques and methods used during this phase aim to extract data from artifacts, such as class diagrams and source code.
- **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 in order to create the **SPL**.

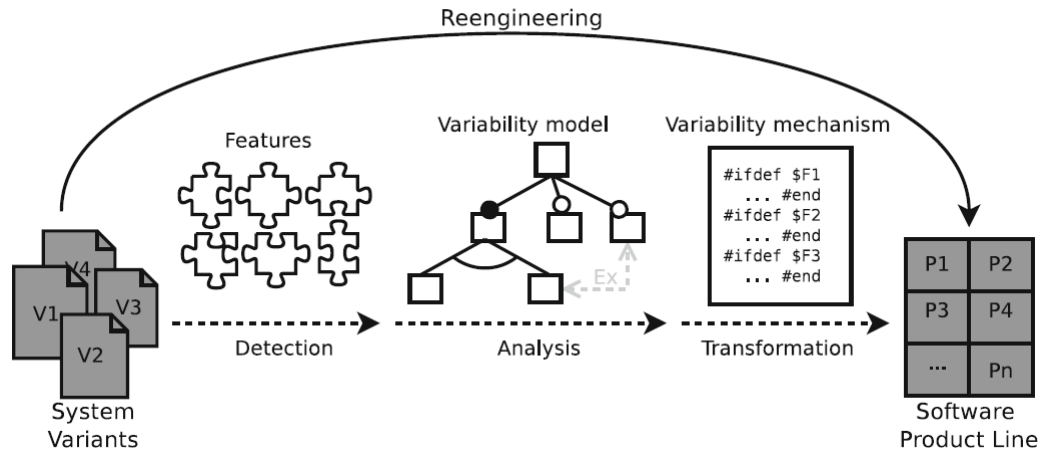
During the first two phases of this process, feature retrieval techniques are used to retrieve the variabilities and commonalities of the products.

2.1.2.1 Feature Retrieval Techniques

Based in **Assunção et al. (2017)**, we classified the feature retrieval strategies in three groups: Static Analysis, Information Retrieval and Support Strategies.

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

Figure 2 – SPL Re-engineering process.



Source: (ASSUNÇÃO et al., 2017)

used source code possesses low coupling and high cohesion. There are a lot of different Static Analysis techniques, from which we highlight:

- Clustering: group a set of objects (*e.g.*, features) based on their similarities in groups called clusters (JAIN; DUBES, 1988);
- Dependency analysis: leverages static dependencies among program elements. It may be used to validate and describe the interdependence between 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 in 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 give emphasis on:

- Formal Concept Analysis (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 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** re-engineering process. Heuristics are proposed by many authors (**RUBIN; CHECHIK, 2012b**) (**BÉCAN et al., 2013**). They are used alongside other techniques to improve the 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.

2.1.3 Feature Model

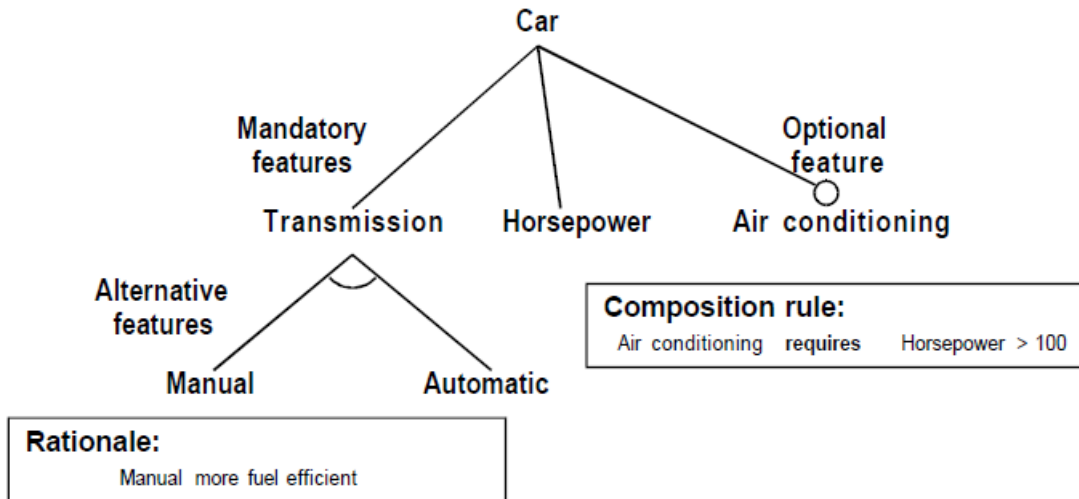
Feature model is one of the most used mechanisms to represent the **SPL** variability (**CLEMENTS; NORTHROP, 2002**). To the best of our knowledge the first feature model notation, Feature-Oriented Domain Analysis (**FODA**) was presented by **Kang et al. (1990)**. The feature model is represented in 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** realisation or not based on the stakeholders choice;
- Alternative *xor-group*: this features are always part of a group where only one of them must be part of the product realisation;
- 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 products realisations based of 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 showed.

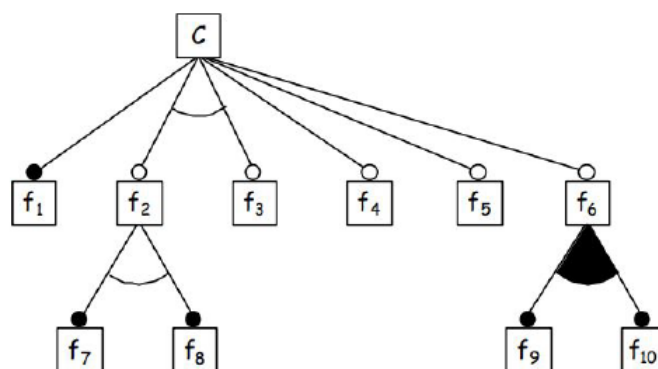
Figure 3 – A feature model of a car.



Source: (KANG et al., 1990)

Other notations, such as Generative Programming presented by Czarnecki e Eisenacker (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*.

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



Source: (CZARNECKI; EISENECKER, 1999)

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 with software development.

Despite its definition, some aspects related with software process may be considered its main areas of investigation. As presented by [Fuggetta e 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, 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 follow different strategies. There is the traditional software development process, which give 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 Chapter Lessons

In this chapter, we explored main concepts of our work. The [SPL](#), [SPLE](#) and [SPL](#) Re-engineering process concepts are important for understanding the impact of our proposal.

Nonetheless, the feature model and feature retrieval techniques 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 different kind of techniques that can be used for different scenarios. Lastly, software process concept is important because we are designing our own process based on approaches found in the literature.

3 RELATED WORK

In this chapter we describe and discuss a systematic literature review performed to search for studies related to ours. Section 3.1.1 describes the protocol used during the review, including the scope and objective in Section 3.1.1.1, the research questions in Section 3.1.1.3, our selection process in Section 3.1.1.4, the criteria used in Section 3.1.1.5 and Section 3.1.1.6. The conduction of the review is detailed in Section 3.1.2 and the related work selected is discussed in Section 3.2.

3.1 A Systematic Literature Review

According to Kitchenham et al. (2009), a Systematic Literature Review (SLR) is a method to examine in-depth and describe the methodology and the results of the primary studies. It provides a complete summary of a research area, identifying most relevant studies and reviewing their contributions. It differs from other methods, such as a Systematic Mapping Study (SMS) (PETERSEN et al., 2008), because it has more specific research questions related to a study giving a deeper analysis about the scope being investigated, while SMS gives a wider view.

3.1.1 Planning Review

This SLR was adapted from the SMS performed by Assunção et al. (2017), which mapped processes for SPL re-engineering. We also wanted to retrieve this proposals, however, we want more specific processes such as flexible and adaptable processes for different scenarios. For this reason, we decided to extend Assunção et al. (2017) SMS adding more criteria, and applying our own Quality/Assessment Criteria (QA) to select more specific works. We also replicate their protocol in proposal of the years 2016 and 2017 because those studies were not part of the Assunção et al. (2017) SMS protocol. To that, we followed the SLR process proposed by Kitchenham et al. (2009).

3.1.1.1 Scope and Objective

In our SLR we were looking for the main contributions in the SPL re-engineering process area. The main objective of the SLR is to make a deep and detailed analysis in this kind of proposals.

3.1.1.2 Question Structure

The research questions were structured according to four criteria:

- *Population*: Research in SPL re-engineering process;
- *Intervention*: Processes containing guidelines for the SPL re-engineering;

- *Outcome*: The expected results are flexible or adaptable processes containing guidelines that are used for **SPL** re-engineering;
- *Context*: Identify which flexible or adaptable processes containing guidelines for **SPL** re-engineering process had been proposed.

3.1.1.3 Research Questions

To compare the retrieved studies with our work, we created three Research Question (RQ)s. These RQs will be used to obtain the main objective of the **SLR**, which is to make a detailed analysis in studies that propose a flexible or adaptable process containing guidelines for **SPL** re-engineering. The RQs are as follows:

RQ1. What are the strategies and artifacts used for the proposals?

RQ2. What are the main contributions of these proposals for **SPL** re-engineering?

RQ3. Are these proposals applicable in more than one scenario?

3.1.1.4 Search Process

To avoid rework we used the 119 studies mapped in Assunção et al. (2017) SMS for the years prior to 2016. For the years 2016 and 2017 we re-applied Assunção et al. (2017) protocol adding our criteria. We used similar databases as them, which included the ACM Digital Library, IEEE Xplore, Springer Link, Scopus, Science Direct and Google Scholar.

We also used the same search string as them, which is showed in Table 1. We adapted the string for each database as the database search engine allowed.

Some databases have a limit of character for the search string. In those cases, we used the string showed in Table 2.

3.1.1.5 Inclusion and Exclusion Criteria

The Inclusion Criteria (IC) and Exclusion Criteria (EC) are used during a **SLR** to remove or maintain studies in the **SLR** analysis. Some of our criteria were taken from the Assunção et al. (2017) SMS, as we intended to apply these criteria in studies of 2016 and 2017. We also created our own EC, and applied them in the mapped studies of (ASSUNÇÃO et al., 2017) and in the studies of 2016 and 2017. To be included, the study should answer Yes for all IC and No for all EC. The criteria of our **SLR** are:

IC1. The study must propose a process for **SPL** re-engineering;

IC2. The proposed process must present some kind of flexibility/adaptability (*e.g.*, use of different artifacts or techniques);

Table 1 – Base Search String.

Search String
(“feature location” OR “concept location” OR “concern location” OR “feature mining” OR “feature identification” OR “feature mapping”) AND (reengineering OR refactoring OR reconstruction OR migration OR migrating OR evolution OR legacy OR restructuring OR “re-engineering” OR “re-structuring”) AND (“application engineering” OR “commonality” OR “core asset” OR “domain analysis” OR “domain engineering” OR “feature analysis” OR “feature based” OR “feature diagram” OR “feature model” OR “feature modeling” OR “feature oriented” OR “highly-configurable system” OR “process family” OR “product family” OR “product line” OR “product line engineering” OR “software family” OR “software product family” OR “software product line” OR “software reuse” OR “SPL” OR “variability” OR “variability analysis” OR “variability management” OR “variability modeling” OR “variability-intensive system” OR “variant” OR “variation” OR “variation point”)

Source: ([ASSUNÇÃO et al., 2017](#))

Table 2 – Reduced search string.

Search String
(“feature location” OR “concept location” OR “concern location” OR “feature mining”) AND (reengineering OR refactoring OR reconstruction OR migration OR migrating) AND (“product line” OR “product-line” OR “product family” OR “product family”)

Source: ([ASSUNÇÃO et al., 2017](#))

- IC3.** The study must be written in English;
- IC4.** The study must be available in electronic format;
- EC1.** The proposed process does not include the detection and analysis phases of the [SPL](#) re-engineering;
- EC2.** The study must propose/use at least two different techniques for feature extraction;
- EC3.** The study is not a primary study;
- EC4.** The main contribution of the study is not a process for [SPL](#) re-engineering
- EC5.** The study does not contain some type of evaluation: examples, case study, experiment, or proof of concept. The evaluation must contain some kind of analysis, showing the achieved results.

3.1.1.6 Quality Assessment Criteria

To rank the retrieved studies, and also to apply a final exclusion to not relevant studies we defined five [QA](#). For each of them, the following scoring was used: Y (yes) =

1; P (partially) = 0.5, N (no) = 0. Hence, the total score, sum of five questions, could result in: 0 to 1.0 (very poor); 1.5 or 2.5 (poor); 2.5 or 3.0 (fair); 3.5 or 4.0 (good) and 4.5 or 5.0 (excellent). In order to grade each study, we considered the following criteria:

QA1. Is the proposed process applicable in more than one scenario?

Evaluation - Y: the proposed process provides evidence of its applicability in more than one scenario; P: the proposed process appears to be applicable in more than one scenario; N: the proposed process is not applicable in more than one scenario;

QA2. Does the proposed process provide some flexibility with regard to the artifacts used?

Evaluation - Y: the proposed process provides/recommends non mandatory artifacts and allows the use of others; P: the proposed process provides/recommends non mandatory artifacts; N: the proposed process provides/recommends only mandatory artifacts;

QA3. Does the proposed process provide some flexibility with regard to the feature retrieval techniques used?

Evaluation - Y: the proposed process provides/recommends non mandatory techniques for feature retrieval and allows the use/combination of others; P: the proposed process provides/recommends non mandatory techniques for feature retrieval; N: the proposed process provides/recommends only mandatory techniques for feature retrieval;

QA4. Does the proposed process provide some or recommends the use of guidelines to support its execution?

Evaluation - Y: the proposed process provides/recommends formalized guidelines to help its execution; P: the proposed process provides/recommends some kind of guideline to help its execution; N: the proposed process does not provide/recommend the use of guidelines;

QA5. Does the study present some kind of analysis, showing the achieved results?

Evaluation - Y: the study presents some kind of analysis or shows the achieved results; P: only a summary of the achieved results is presented; N: neither analysis nor results are specified.

3.1.1.7 Selection Process

Our selection process is divided in 5 steps:

1. Search databases: Initially, strings were generated by means of the selected keywords and synonyms and the selection of studies of 2016 and 2017 was performed;

2. Eliminate redundancies: There were some redundancies since same studies were returned by different search engines. In this step, we eliminated those;
3. Intermediate selection: The title and the abstract (reading the introduction and conclusion when necessary) were read for each study returned by the search engines and studies were excluded based on the criteria mentioned in Section 3.1.1.5;
4. Merge: The selected studies of 2016 and 2017 were merged with studies mapped by (ASSUNÇÃO et al., 2017);
5. Detailed selection: The remaining studies were full analyzed and the IC and EC were applied again;
6. Final selection: Based on the quality criteria (see Section 3.1.1.6), we evaluated the quality of studies that were read in the Final Selection step. We ranked the studies based on the criteria and analyzed them to answer our RQs.

3.1.2 SLR Conduction

The SLR was conducted between July/2017 and September/2017 according to the selection process presented in Section 3.1.1.7. In total, 354 works were found from the years 2016 and 2017. We merged the 119 works mapped by (ASSUNÇÃO et al., 2017), and 13 studies remained for the QA application. Table 3 shows the number of returned works for each database (between 2016 and 2017).

Table 3 – Retrieved studies by database.

Database	Number of Studies
ACM	2
IEE	1
Google Scholar	74
Science Direct	133
Scopus	124
SpringerLink	20
Total	354

Source: Author.

The summary of the selection process can be seen in Table 4. We analyzed the total of 354 studies from 2016 and 2017, plus the 119 studies mapped by (ASSUNÇÃO et al., 2017). All studies were analyzed by two researchers and their results were merged. If they had disagreed in something a third research would decide whether to include the work or not. In the end of the process, a total of 13 studies were analyzed and ranked according to QA (Table 5), and we selected 5 studies (ranked as good or excellent) to analyze and include as a work related to ours. This small number of studies selected may indicate that there are few works similar to ours.

Table 4 – Summary of the selection process.

Step	Remaining Studies
Search Databases	354
Eliminate Redundancies	306
Intermediate Selection	26
Merge	145
Detailed Selection	13
Final Selection	5

Source: Author.

Table 5 – Quality assessment result.

Study	QA1	QA2	QA3	QA4	QA5	Rank
Acher et al. (2012)	P	P	P	Y	Y	Good
Acher et al. (2013)	Y	Y	N	P	Y	Good
Araar e Seridi (2016)	Y	N	N	Y	Y	Fair
Assunção et al. (2016)	Y	N	N	P	Y	Fair
Assunção (2017)	Y	N	N	P	Y	Fair
Bécan et al. (2013)	Y	P	P	P	Y	Good
Kulesza et al. (2007)	P	N	N	Y	Y	Fair
Kumaki et al. (2012)	P	P	N	P	Y	Fair
Martinez et al. (2015)	P	Y	Y	Y	Y	Excellent
Merschen et al. (2011)	N	P	P	P	Y	Fair
Polzer et al. (2012)	N	P	P	P	Y	Fair
Santos et al. (2013)	Y	P	P	Y	Y	Good
Tang e Leung (2015)	Y	N	P	P	Y	Fair

Source: Author.

3.2 Discussing Related Work

In this section, we discuss each one of the five related works selected during the SLR process. The works were analyzed to compare their main contributions and their flexibility as well. We have focused on the following topics:

- Artifacts and strategies used for feature retrieval (RQ1);
- The main contribution of each study (RQ2);
- Applicability in different scenarios (RQ3).

3.2.1 Artifacts and Strategies for Feature Retrieval

We summarize the artifacts used for each process proposed in Table 6, in which we can see that most of the studies focused on design models and requirements as input artifacts. However, ([MARTINEZ et al., 2015](#)) ([SANTOS et al., 2013](#)) had used also

source code, giving their process more flexibility in this topic. However, the works present some lack of flexibility. For instance, even the most flexible (MARTINEZ et al., 2015) (SANTOS et al., 2013), require the use of specific artifacts types (*e.g.*, requirements list), ignoring the use of some artifacts (*e.g.*, reference architecture). On the other hand, our process intends to make use of all available artifacts, to obtain the highest precision when retrieving features.

With regard to the strategies used for feature retrieval, most of the studies make use of Expert-Driven Extraction. However, this strategy is used alongside another, as Table 7 shows, (ACHER et al., 2012) (ACHER et al., 2013) (BÉCAN et al., 2013) (MARTINEZ et al., 2015) used static analysis techniques, such as structural similarity, and (SANTOS et al., 2013) used information retrieval techniques to complement the expert-driven method. Similar to artifacts, our process intends to make use of all possible strategies. In the selected studies, however, usually one strategy is used alongside Expert-Driven as a support strategy.

Table 6 – Artifacts used for each study.

Study	Artifacts			
	Design Models	Domain Information	Requirements	Source Code
Acher et al. (2012)	✓	✓		
Acher et al. (2013)	✓		✓	
Bécan et al. (2013)			✓	
Martinez et al. (2015)	✓		✓	✓
Santos et al. (2013)	✓		✓	✓

Source: Author.

Table 7 – Retrieval strategies used for each study.

Study	Strategies		
	Expert Driven	Static Analysis	Information Retrieval
Acher et al. (2012)		✓	
Acher et al. (2013)	✓	✓	
Bécan et al. (2013)		✓	
Martinez et al. (2015)	✓	✓	
Santos et al. (2013)	✓		✓

Source: Author.

3.2.2 Studies Main Contribution

Considering the main contributions of the proposals, Acher et al. (2012) had proposed a retrieval process focusing on plugin-based systems. Their process automati-

cally 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.

Acher et al. (2013) work presents a process, alongside a language and tool to extract the variability of the products. They used products specification and description to extract and synthesize a feature model. These products artifacts may be of different types, however the techniques used its always the merging of products descriptions using structural similarity.

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. Their techniques may be applicable without prior knowledge, reducing the effort in comparison to other proposals.

The main contribution of the work presented in Martinez et al. (2015) is a generic and extensible framework for SPL re-engineering. Their main goal is to reduce the high investment required to adopt the SPLE. The proposed framework may be easily adapted to different artifacts types and integrates state-of-the-art algorithms and visualization paradigms. The work also presents a realisation of the framework, called Bottom-Up Technologies for Reuse (BUT4Reuse).

The last selected work, (SANTOS et al., 2013), proposes to use testing to support the feature retrieval process. They use testing as 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.

Some studies presented similar proposals (BÉCAN et al., 2013) (MARTINEZ et al., 2015), which intend to be generic and extensible. There are also the use of different artifacts as main input, testing (SANTOS et al., 2013) and product descriptions (ACHER et al., 2013). Aside these differences, the main contributions of the studies could be summarized as reduce the effort and guide the users to retrieve the features, which is similar to our main goal.

3.2.3 Applicability in Different Scenarios

Some of the selected works aim to be applicable in different scenarios and had evidenced it with empirical evaluations (ACHER et al., 2013) (BÉCAN et al., 2013) (SANTOS et al., 2013). While Acher et al. (2012) Martinez et al. (2015) only indicate their possible applicability without giving any evidence. Most of these different scenarios, however, are different systems used as main input for the proposals. They do not consider different artifacts of the same systems, or a different team performing the re-engineering

either. The work of [Acher et al. \(2012\)](#) however, is specifically created to be applied only in plugin-based systems. The work of [\(MARTINEZ et al., 2015\)](#) was created with the goal of be extensible in many different scenarios and situations, being the most adaptable/flexible of the selected works.

Comparing to our work, however, the proposals do not include a step/phase to collect information about the experience applying the retrieval process (see Section 4.2.3.3). This information may be used during the re-applicability in a different scenario, reducing the effort and improving precision.

3.3 Chapter Lessons

In this chapter we presented a [SLR](#), with its protocol (Section 3.1.1), the [SLR](#) conduction (see Section 3.1.2) and its results. The results showed that exists a considerable number of works focusing on process for [SPL](#) re-engineering. During this [SLR](#), 354 studies from 2016 and 2017 were analyzed then merged with the 119 studies from the [SMS](#) of [Assunção et al. \(2017\)](#). At the end, 13 studies were analyzed according to [QA](#) and 5 studies were selected to be discussed as a related work in Section 3.2. These 5 selected studies were analyzed to compare their artifacts and strategies used for feature retrieval, main contribution, and applicability in different scenarios. The analysis results indicated that the studies present some kind of flexibility (*e.g.*, use of different type of artifacts), however this flexibility could be improved. Considering the main contribution, all studies intend to reduce the effort of feature retrieval. Lastly, the applicability in different scenarios was evaluated by some studies ([ACHER et al., 2013](#)) ([BÉCAN et al., 2013](#)) ([SANTOS et al., 2013](#)), while others only indicated the possibility ([ACHER et al., 2012](#)) ([MARTINEZ et al., 2015](#)).

4 THE PREPARE, ASSEMBLE AND EXECUTE PROCESS

This chapter introduces the Prepare Assemble and Execute Process for Software Product Line Re-engineering (PAXSPL). The content presented here, describes the relevance of this work and our expectations. In Section 4.1, we present our proposed process. In Section 4.3 we present the guidelines created to support the process. Lastly, in Section 4.4, we present challenges and missing work.

4.1 Process Overview

PAXSPL is a process that assembles and executes a feature retrieval process for SPL re-engineering. Our process takes as an input a set of similar systems that have potential to become SPL. We designed our process to allow adaptation and re-applicability according to different scenarios. This may reduce the cost and the effort of re-applying this process in different or similar scenarios. Our process analyzes data such as team experience, team skills, and products documentation. This information is used to choose most recommended retrieval techniques for a given scenario. These techniques are assembled in different activities of our generic process, generating an assembled process. Then, this process is executed to retrieve features of the product variants. At the end, our process generates a feature retrieval process that is assembled for the specific scenario. This feature retrieval process generates a feature diagram of the future SPL. The information presented in the following sections represents the documentation of our process, which is also available in a repository¹.

4.1.1 Process Structure

The process can be divided in:

- Actors/Roles: roles assign to team members, responsible for executing the activities;
- Artifacts: concrete artifacts used by activities or produced by them;
- Phases: there are three main phases in our process, each one of them contain a set of activities;
- Activities/Sub-process: main activities of the process which may include more activities within;
- Guidelines: a set of mapped strategies and techniques containing information about them and recommended situations.

In the following sub-sections we describe in detail each actor/role, artifacts, phase, activity/sub-process and guidelines.

¹ <<https://github.com/HestiaProject/PAXSPL/>>

4.1.2 Actors/Roles

We established a set of six roles for our process. However, aiming to give more flexibility we encourage the creation of more roles or even the modification of those six. The roles are all optional and they can be performed by more than one team member. Also, one team member may assume more than one role. The six roles defined for our process are:

- **Domain Engineer:** possesses valid knowledge related to an area of human endeavour, an autonomous computer activity, or other specialized discipline. Basically, it must be expert in the software domain;
- **Architect:** has valid knowledge related to software architecture, design and architectural patterns. It is recommended to be the same person(s) who developed the individual systems architecture;
- **Analyst:** specialist in the requirements of the systems being re-engineered. It should be aware of functional and non functional requirements;
- **Developer:** possesses programming languages and technologies knowledge. May be the same person(s) who developed the source code of the individuals systems;
- **Feature Checker:** must know domain glossary, domain constraints, and requirements information. It will be responsible for verifying the features retrieved to look for problems;
- **Feature Retriever:** must have knowledge about feature retrieval strategies and techniques. It is going to apply the selected techniques to identify and extract the features from the systems being re-engineered.

4.1.3 Artifacts

The artifacts of our process can be divided in input and output artifacts. To extend the flexibility in this point, we utilized the feature model classification (Section 2.1.3) to classify the input artifacts in: mandatory, optional and alternative or-group. In this case, mandatory artifacts must be used, optional may be used or not, and in an alternative or-group, at least one artifacts should be used as input. There are many kinds of artifacts to be used, and many variations (*e.g.*, use cases or user stories as requirements artifacts). The team performing the process will have to use the product artifacts possesses by them. In the following sub-sections we give more details about input and output artifacts. We also describe three important artifacts of our process: documentation set, generic and assembled process.

4.1.3.1 Input Artifacts

Input Artifacts are the artifacts collected from the products variants used in the re-engineering process. Some artifacts, such as domain information, may be used either as input or output.

- Source Code: code that implements the software products being re-engineered;
- Requirements: requirements of the software products. Requirements can be described using different ways, such as a list that contains functional and non functional requirements of the system. They can be ordered by priority, cost or preference by the stakeholder;
- Design Models: models designed during the project of the products used in the re-engineering process (*e.g.*, class diagrams, state machine diagrams, activity diagrams);
- Domain Information: artifacts created during the domain engineering, focusing on the development of reusable assets such as reference architecture or reference requirements.

4.1.3.2 Output Artifacts

Output artifacts are the artifacts generated during the process execution. Many of them may also be used as input in some activities.

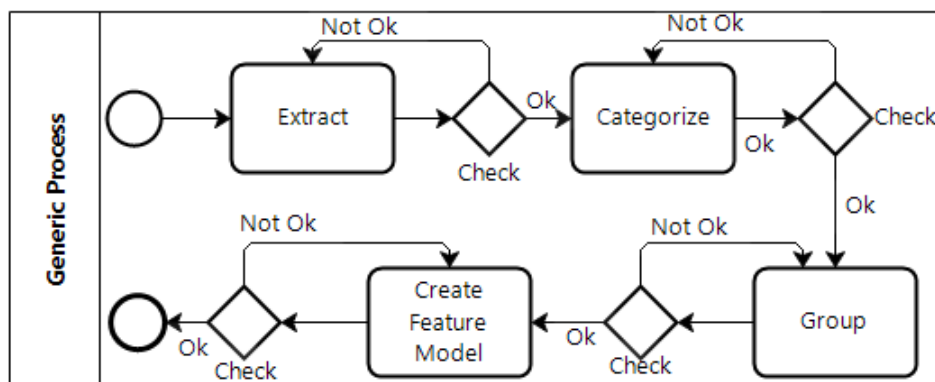
- Feature Artifacts: are generated from the application of the feature retrieval techniques. Each technique may generate different artifacts such as graph representation of features or feature descriptions;
- Variability Reports: reports of the variability of the [SPL](#). Written in natural language, they contain information such as how some features were extracted and why they should be mandatory or not;
- Modified Product Artifacts: product artifacts modified to include feature entry points. These entry points are the location where the feature was discovered in the artifacts. They may be source code comments, use case annotations, etc;
- Team Information Report: report(s) containing information such as team experience, team skills and team knowledge. This document should contain the information of all team members;
- Domain Artifacts: the artifacts shared by all systems of the future [SPL](#). They usually describe domain constraints, domain glossary;

- Artifacts Types Specification: information about artifacts types such as extensions, formats and templates used.
- Development Information: design patterns, architectural patterns, programming patterns, programming and development paradigms and even technological information can be included.

4.1.3.3 Generic and Assembled Processes

The assembled process is the result of the tailoring performed by the team during **Assemble** where techniques are selected and placed into the **Generic Process** according to team's decisions. As showed in Figure 5, the **Generic Process** of PAXSPL contains the four main phases identified by analyzing the process proposed in the literature. During **extract** (see Figure 5), features are identified and feature entry points are added to product artifacts. In the **categorize** activity, features are classified as mandatory, optional, etc. **Group** is when the features are separated according to their relations and dependencies. At last, the feature model is created. After each activity, a check is made to verify if problems or inconsistencies exists. The user may decide how to perform this check activity.

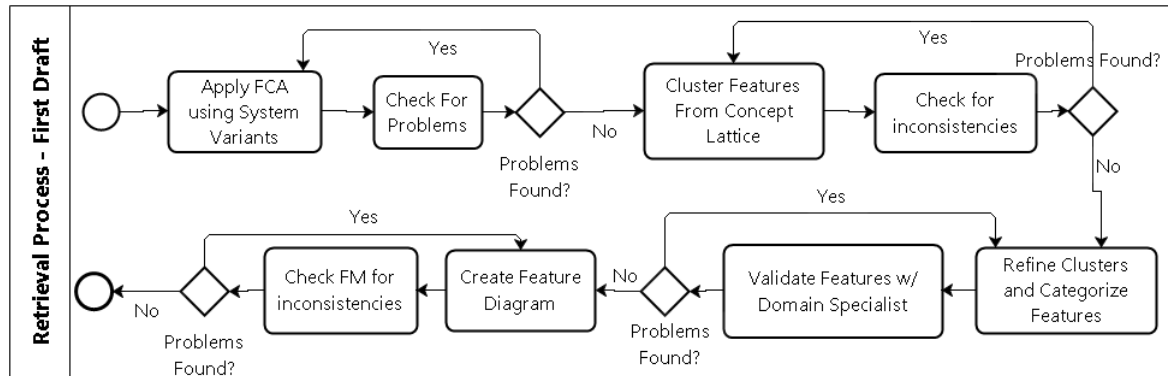
Figure 5 – The generic process for feature retrieval.



Source: Author.

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 6, 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 assign to a specific actor, which means they may be performed by anyone. As Figure 6 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 6 – An example of assembled process.

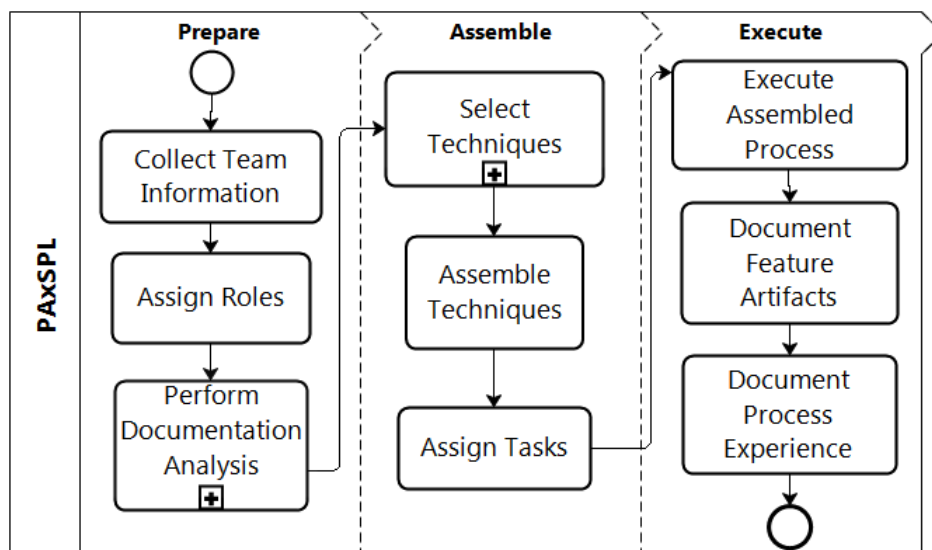


Source: Author.

4.2 Phases and Activities

Our process is divided in three sequential phases, as illustrated in Figure 7: **Prepare**, **Assemble** and **Execute**. During **Prepare**, information is collected and analyzed to prepare artifacts that will help the process assembly. **Assemble** is when the techniques are selected and the retrieval process is created. This retrieval process is performed during **Execute** phase. Also, there is some information collected during this phase such as the process execution experience. We detail the activities of each phase in the following sub-sections. For each activity we give the actors and the artifacts directly related to the activity.

Figure 7 – The PAXSPL process.



Source: Author.

4.2.1 Prepare

In the first phase of our process, information is collected in order to prepare the process assembly. This phase is divided in three main activities: **Collect Team Information**, **Assign Roles**, and **Perform Documentation Analysis**. The latter, **Perform Documentation Analysis** is actually a sub-process that contains activities within.

4.2.1.1 Collect Team Information

Information about the team which will execute the process is collected. This information includes team members experience, team members skills, team members knowledge and preferences of each member. The information is registered in a document, the **Team Information Report** (see Appendix A).

- Actors: Any;
- Optional Inputs: Project plan, business organization chart, team members profile, any other team member relevant information;
- Outputs: Team Information Report.

4.2.1.2 Assign Roles

Roles are assigned based on the information collected on the previous activity. Usually, team members that possess more knowledge about the software domain may assume the role of **Domain Engineer**, members that worked in the individual systems analysis and requirements specifications may assume the role of **Analyst**, and so on. These roles are related with the next activity, which is **Perform Documentation Analysis**.

- Actors: All;
- Optional Inputs: Project plan, business organization chart, team members profile, any other team member relevant information;
- Mandatory Inputs: Team Information Report;
- Outputs: Team Information Report updated.

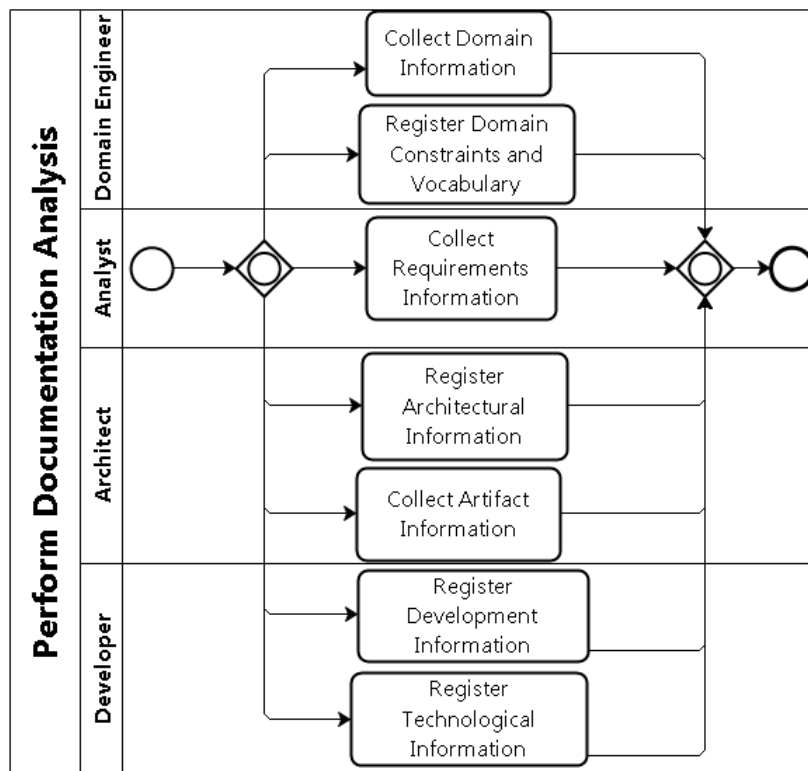
4.2.1.3 Perform Documentation Analysis

During this sub-process the re-engineering documentation is collected and compiled. All activities of this sub-process are optional and they can be performed at the same time. Collected information may include domain information, requirements information of the products, and information about architecture, technologies and artifacts. All these information, alongside the team information report, are added to the **Documentation**

Set. Figure 8 gives an overview of this sub-process, which is detailed in the following sub-sections.

- Actors: All;
- Outputs: Documentation Set updated.

Figure 8 – Documentation analysis sub-process.



Source: Author.

4.2.1.3.1 Collect Domain Information

This is an optional activity and will or will not be performed based on the need of its outputs artifacts. In other words, if domain information such as reference architecture or reference requirements are needed, this activity may be performed. During this activity, domain information is collected and registered. This information can be used as an input for some extraction techniques. According to [Assunção et al. \(2017\)](#), domain artifacts may contain information such as products description, user comments, documentation of systems in specific domain, and domain analysis.

- Actors: Domain Engineer;

- Alternative Inputs: Reference architecture, reference requirements, domain realisation, domain tests;
- Outputs: A set of domain artifacts.

4.2.1.3.2 Register Domain Constraints and Vocabulary

This is also an optional activity, where a list of constraints and terms related to the systems domain is collected and registered. These constraints may be collected in the system business rules or even non functional requirements.

- Actors: Domain Engineer;
- Alternative Inputs: Reference architecture, reference requirements, domain realisation, domain tests;
- Outputs: Domain glossary and constraints list.

4.2.1.3.3 Collect Requirements Information

This is an optional activity where requirements information is collected and registered. This information should be collected from all systems being re-engineered, however, sometimes the systems do not have this kind of documentation. The ideal scenario when this documentation is missing is to create it.

- Actors: Analyst;
- Alternative Inputs: requirements list, use cases, business rules, and other requirements artifacts;
- Outputs: Requirements specification.

4.2.1.3.4 Register Architectural Information

This is other optional activity where architectural information is collected and registered. This information also may be missing from some products being re-engineered. The ideal scenario would be the identification of this missing information.

- Actors: Architect;
- Alternative Inputs: Class diagrams, state machine diagrams, activity diagrams, or any other design model of the individual systems;
- Outputs: Development information.

4.2.1.3.5 Collect Artifact Information

This is another optional activity where information about artifacts types (*e.g.*, extensions, formats, structures, etc) is collected and registered. This information may be used to decide which extraction techniques should be considered or excluded from the assemble process.

- Actors: Architect, Analyst, Developer;
- Alternative Inputs: Class diagrams, state machine diagrams, requirements specification, or any other artifact that contains relevant information about the individual systems;
- Outputs: Artifacts type specification.

4.2.1.3.6 Register Development Information

This is another optional activity where information about the developed products will be collected and registered. This information will also be a decision point when choosing the techniques. This information may include programming patterns, programming and development paradigms.

- Actors: Developer;
- Mandatory Input: source code;
- Optional Inputs: class diagrams, activity diagrams, state machine diagrams, or any other artifacts containing information about the products development;
- Outputs: artifacts type specification, development information.

4.2.1.3.7 Register Technological Information

This is another optional activity where information about technologies used in each product are collected and registered. This information may exclude the use of some techniques, or even give more priority to others.

- Actors: Developer;
- Mandatory Input: source code;
- Optional Inputs: reference architecture;
- Outputs: development information.

4.2.2 Assemble

In this phase, the information collected is analyzed and techniques are selected and assembled into the generic process. This phase could be considered the tailoring of the feature retrieval process. This tailoring is done by selecting techniques and placing them into each phase of the generic process. The activities of this phase are detailed in the following sub-sections.

4.2.2.1 Select Techniques

During this activity the information collected during **Prepare** is analyzed to help the selection of techniques for the feature retrieval.

First, a candidate technique is selected to be analyzed, as illustrated in Figure 9. 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 analyzed considering three points: practical examples, which are examples of the technique use found in the literature; comparison among technique recommended scenarios with current scenario, also found in the literature. The users should analyze whether their scenario contain some similarity in comparison with the scenarios where the technique had being used or not; and check available product artifacts with technique artifacts, for instance whether a technique uses requirements artifacts they should be available to use.

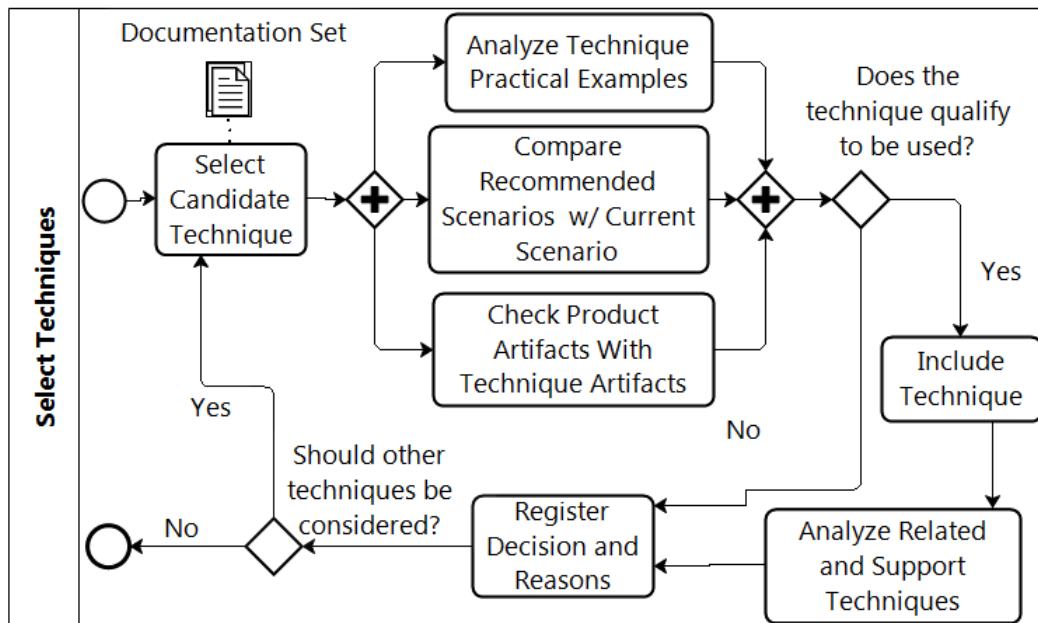
Based on the analysis performed consider those three points, the user must decide to use the techniques or not. If the technique qualify to be used, it is included as a selected technique, then related and support techniques are analyzed to decided whether they may be candidate techniques. Then, the decision of including or not the technique must be registered along with its reasons. If another technique is considered a candidate, then the process repeats.

- Actors: Any;
- Mandatory Input: Documentation set;
- Optional Inputs: Any artifact the team decide to use;
- Outputs: A feature model showing the selected techniques.

4.2.2.2 Assemble Techniques

The techniques chosen are assembled inside our generic process. The assembled process will be customized according to the scenario where our process has been applied, giving our process flexibility to be applied in different situations. In each activity of the **Generic Process** one or more techniques must be placed.

Figure 9 – Select techniques sub-process.



Source: Author.

- Actors: All;
- Mandatory Input: Selected techniques;
- Optional Inputs: a draft of the assembled process;
- Outputs: the assembled process for feature extraction.

4.2.2.3 Assign Tasks

Each member of the team should receive a task to perform during the process execution. One member can perform more than one task and a task can be performed by multiple members. The team is totally free to divide the tasks, create pairs or sub-teams to perform task, or use any other kind of strategy.

- Actors: All;
- Mandatory Input: the assembled process for feature extraction;
- Optional Inputs: documentation set;
- Outputs: the assembled process updated.

4.2.3 Execute

During this phase, the assembled process is executed and the feature artifacts are collected. Also, reports are created to document the experience of the process execution. The activities of this phase are detailed below.

4.2.3.1 Execute Assembled Process

The assembled process is executed to retrieve the features from the systems being re-engineered.

- Actors: Any;
- Mandatory Input: the assembled process for feature extraction;
- Outputs: feature artifacts, feature model.

4.2.3.2 Document Feature Artifacts

Feature artifacts are documented in a structured way. These artifacts will depend on the techniques selected.

- Actors: Any;
- Mandatory Input: the assembled process for feature extraction, feature artifacts, feature model.
- Outputs: feature artifacts formalized.

4.2.3.3 Document Process Experience

Reports are created to document the experience of the process execution. These reports may be used in future re-executions of the process, reducing cost and effort. Experience reported may include: mainly difficulties during the execution, important decision factors that led to chose a technique for extraction, and any information the team thinks is relevant for future executions.

- Actors: Any;
- Optional Input: lessons learned during the process execution;
- Outputs: process execution report.

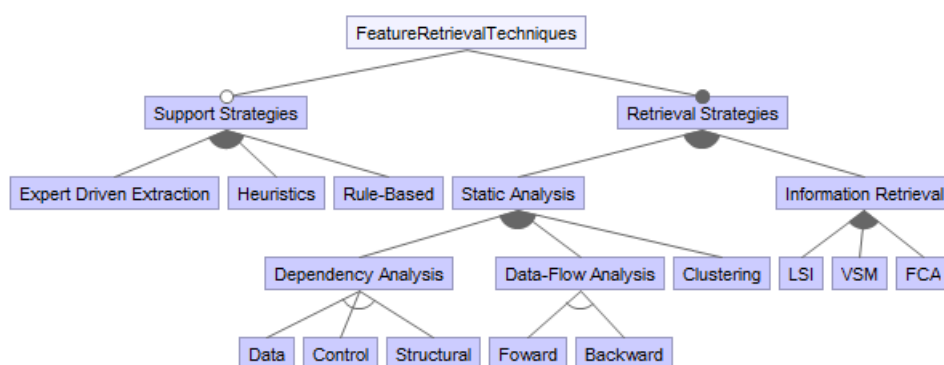
4.3 Guidelines

We created a guidelines documentation to help choosing 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](#), re-engineering, variability management, a feature model notations. This information is important for user with low experience with [SPL](#) re-engineering and may decrease their difficulty when performing [PAXSPL](#). Within our guidelines we also included a checklist (see Section 4.3.4) to give support during the process execution.

With this kind of documentation, we intend to give as much help as possible for the team make the best decisions when selecting the techniques for feature retrieval. We also created a feature model of Feature Retrieval Techniques that formalizes possible techniques and makes possible to calculate how many different combinations can be chosen. We grouped the techniques based on their strategy, as shown in Figure 10:

- Static Analysis: Dependency Analysis and its variations, Data-Flow analysis and its variations, and Clustering;
- Information Retrieval: [LSI](#), [VSM](#) and [FCA](#);
- Support Strategies: Rule-Based Techniques, Expert-Driven Extraction and Heuristics.

Figure 10 – A feature model of retrieval techniques.



Source: Author.

4.3.1 Static Analysis

Static analysis is done by analyzing a program without its execution. Information analyzed may include structure and static artifacts. Static analysis techniques are recommended when analyzing source code. Furthermore, is recommended that the used source

code possesses low coupling and high cohesion to improve the precision. The use of static analysis is also recommended alongside at least one information retrieval technique such as formal concept analysis.

4.3.1.1 Clustering

The definition of clustering is to group elements, in this case features, based on their dependencies in groups called clusters.

- Variations: (i) Agglomerative Hierarchical Clustering (AHC): A “bottom up” approach where each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. (ii) Divisive Hierarchical Clustering (DHC): A “top down” approach where all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.
- Priority Order: Group >Extraction >Categorize.
- Inputs: Source Code.
- Outputs: Feature tree, feature clusters, dendrogram tree.
- Examples: (ALVES et al., 2008) (EYAL-SALMAN; SERIAI; DONY, 2014) (CHEN et al., 2005) (DAMAŠEVIČIUS et al., 2012) (KELLY et al., 2011) (NÖBAUER; SEYFF; GROHER, 2014b) (RUBIN et al., 2012) (WESTON; CHITCHYAN; RASHID, 2009) (BÉCAN et al., 2013).
- Tools: Cluster 3.0, Pycluster, C Clustering Library².
- Related Techniques: FCA.
- Recommended Situations: Clustering is highly recommended in products that possesses high level of dependencies among feature implementations. Besides that, a good documentation is not required when applying this technique.
- Not Recommended Situations: As a static analysis technique, clustering may be unable to find all elements related to the same feature when applied in source code where the implementation of a feature is spread across several modules.

4.3.1.2 Dependency Analysis

Can be defined as the act of leverages static dependencies among program elements. Can be used to validate and describe the interdependence among elements.

² <<http://bonsai.hgc.jp/~mdehoon/software/cluster/software.htm>>

- Variations: (i) Data Analysis. (ii) Control Analysis. (iii) Structural Dependency: A metric that represents how much of the structural context of a feature can be mapped from a set of program elements.
- Priority Order: Extraction >Categorize >Group.
- Inputs: Source Code.
- Outputs: Dependence graph.
- Examples: (ALI et al., 2011) (NÖBAUER; SEYFF; GROHER, 2014a) (KLATT; KROGMANN; SEIDL, 2014) (LINSBAUER et al., 2014).
- Tools: Dependency Finder³, Java Dependency Analysis Tool, eDepend.
- Related Techniques: FCA.
- Recommended Situations: To perform Dependency Analysis is almost mandatory that the products have high level of dependencies among feature implementations. Besides that, a good documentation is not required when applying this technique.
- Not Recommended Situations: As a static analysis technique, dependency analysis may be unable to find all elements related to the same feature when applied in source code where the implementation of a feature is spread across several modules.

4.3.1.3 Data Flow Analysis

Data Flow Analysis may be defined as the collection of information about possible values calculated at different points of a software system. This information is used to determine in which parts of that program a particular value might propagate.

- Variations: (i) Forward Analysis: Calculates for each program point the set of definitions that may potentially reach this program point. (ii) Backward Analysis: Calculates for each program point the variables that may be potentially read afterwards before their next write update.
- Priority Order: Extraction >Categorize >Group.
- Inputs: Source Code.
- Outputs: Code fragments related to a feature.
- Examples: (BAYER et al., 2004).
- Tools: InputTracer⁴.

³ <<http://depfind.sourceforge.net/>>

⁴ <<http://ieeexplore.ieee.org/document/6392112/>>

- Related Techniques: Clustering.
- Recommended Situations: As any static analysis technique, data flow analysis may be apply in a satisfactory way, when source code is well written. Also, better results can be reached when source code possesses high level of dependencies among feature implementations.
- Not Recommended Situations: If the source code possesses a high variable flow data flow analysis may have uncertain results.

4.3.2 Information Retrieval

This strategy collects and analyzes information in artifacts considering text structure, similarity, etc. Information retrieval techniques commonly use documents written in natural language, generally used in requirements artifacts. However, they can also be used in source code, but to do that, either source code and requirements must have meaningful names. Is also recommend to use information retrieval techniques alongside at least one static analysis technique, such as clustering.

4.3.2.1 Formal Concept Analysis

FCA is a mathematical method that provides a way to identify “meaningful groupings of objects that have common attributes”.

- Variations: (i) Clarified: The context does not have duplicate rows or columns. (ii) Reduced: No context rows can be expressed as intersection of other rows. The lattice of such context is meet-reduced. No context columns can be expressed as intersection of other columns. The lattice of such context is join-reduced.
- Priority Order: Extraction >Categorize >Group.
- Inputs: Source Code, Requirements, Design Models, or any textual artifact.
- Outputs: Concept lattice.
- Examples: (EISENBARTH; KOSCHKE; SIMON, 2001) (YANG; PENG; ZHAO, 2009) (EYAL-SALMAN; SERIAI; DONY, 2014) (EYAL-SALMAN; SERIAI; DONY, 2013a) (AL-MSIE'DEEN et al., 2013) (AL-MSIE'DEEN et al., 2012) (XUE; XING; JARZABEK, 2012) (MAAZOUN; BOUASSIDA; BEN-ABDALLAH, 2014) (MAÂZOUN; BOUASSIDA; BEN-ABDALLAH, 2014) (MEFTEH; BOUASSIDA; BEN-ABDALLAH, 2014).
- Tools: We found an extensive list of **FCA** supporting tools⁵.

⁵ <<http://www.upriss.org.uk/fca/fcasoftware.html>>

- Related Techniques: [LSI](#), Clustering.
- Recommended Situations: [FCA](#) is recommended when program elements (*e.g.*, classes, methods, etc.) have meaningful names (*e.g.*, “attribute” instead of “atr” or “home” instead of “hm”). Besides that, is highly recommended to use this technique in products well documented.
- Not Recommended Situations: As an information retrieval technique, [FCA](#) cannot achieve quality results when applied to products with no documentation and no meaningful identifiers names. For that reason it is not recommended in those situations.

4.3.2.2 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.

- Variations: (i) Semantic Hashing: Documents are mapped to memory addresses by means of a neural network in such a way that semantically similar documents are located at nearby addresses.
- Priority Order: Extraction >Categorize >Group.
- Inputs: Source Code, Requirements, Design Models, or any textual artifact.
- Outputs: Occurrence matrix.
- Examples: ([AL-MSIE'DEEN et al., 2013](#)) ([XUE; XING; JARZABEK, 2012](#)) ([EYAL-SALMAN; SERIAI; DONY, 2013b](#)) ([EYAL-SALMAN; SERIAI; DONY, 2013a](#)) ([AL-MSIE'DEEN et al., 2012](#)) ([MAAZOUN; BOUASSIDA; BEN-ABDALLAH, 2014](#)) ([MAÂZOUN; BOUASSIDA; BEN-ABDALLAH, 2014](#)) ([ALVES et al., 2008](#)).
- Tools: Ultimate Keyword Hunter⁶, Infomap NLP Software⁷, LSA of Colorado University⁸
- Related Techniques: [FCA](#), Clustering.
- Recommended Situations: [LSI](#), same as [FCA](#) is recommended when program elements (*e.g.*, classes, methods, etc.) have meaningful names (*e.g.*, “attribute” instead of “atr” or “home” instead of “hm”). Besides that, is highly recommended to use this technique in products well documented.

⁶ <<http://ultimatekeywordhunter.com/>>

⁷ <<http://infomap-nlp.sourceforge.net/>>

⁸ <<http://lsa.colorado.edu/>>

- Not Recommended Situations: As an information retrieval technique, [LSI](#) cannot achieve quality results when applied to products with no documentation and no meaningful identifiers names. For that reason it is not recommended in those situations.

4.3.2.3 Vector Space Model

[VSM](#) is an algebraic model for representing text documents in a way where the objects retrieved are modeled as elements of a vector space.

- Variations: N/A.
- Priority Order: Extraction >Categorize >Group.
- Inputs: Source Code, Requirements, Design Models, or any textual artifact.
- Outputs: Vectors representing the objects retrieved.
- Examples: ([KUMAKI et al., 2012](#)) ([EYAL-SALMAN; SERIAI; DONY, 2013b](#)) ([ALVES et al., 2008](#)).
- Related Techniques: [FCA](#), Clustering.
- Recommended Situations: Same as [LSI](#) and [FCA](#), [VSM](#) is recommended when program elements (*e.g.*, classes, methods, etc.) have meaningful names (*e.g.*, “attribute” instead of “atr” or “home” instead of “hm”). Besides that, is highly recommended to use this technique in products well documented.
- Not Recommended Situations: The use of [VSM](#) has some limitations which may be considering when selecting this technique, such as: (i) Long documents are poorly represented because they have poor similarity values; (ii) Search keywords must precisely match document terms: word sub-strings might result in a “false positive match”; (iii) Semantic sensitivity: documents with similar context but different term vocabulary won’t be associated, resulting in a “false negative match”; (iv) The order in which the terms appear in the document is lost in the vector space representation.

4.3.3 Support Techniques

These techniques are used alongside information retrieval or static analysis techniques. Their goal is to improve the results of some techniques. Support techniques may never be performed alone.

4.3.3.1 Expert-Driven

This technique is based on knowledge and experiences of specialists, such as domain engineers, software engineers, stakeholders, etc. This may include the addition of techniques that are not in this process documentation, if the specialist decide to use them. To apply the expert-driven technique is highly recommended to have a team with skills and knowledge involving the re-engineering process of [SPL](#). We also recommend to use Expert-Driven as a support technique along Static Analysis and Information Retrieval.

4.3.3.2 Heuristics

Heuristics are proposed approaches that use a practical method not guaranteed to be perfect, but sufficient to obtain immediate goals. We mapped a set of heuristics proposed by different works in the [SPL](#) re-engineering area, we give some examples:

- H1** Filtering: Filter elements that do not contribute directly to the feature retrieved ([RUBIN; CHECHIK, 2012b](#)).
- H2** Score Modification: Rank elements based on their lexical and syntactical properties, eliminating not relevant elements ([RUBIN; CHECHIK, 2012b](#)).
- H3** Compare: Calculates the similarity degree for each pair of input model elements ([RUBIN; CHECHIK, 2012a](#)).
- H4** Match: Use empirical similarity thresholds and analyze a pair of model elements, returning pairs that are considered similar in a match result ([RUBIN; CHECHIK, 2012a](#)).

We decided to include heuristics in our process documentation because we think they can be combined with any other feature retrieval technique, giving our process more flexibility.

4.3.3.3 Rule-based

Similar to heuristics, this techniques uses a set of rules is to guide and help whoever is performing the feature extraction. One example may be seen in the work of [Mu, Wang e Guo \(2009\)](#), where these rules are applied in source code and requirements to extract the features alongside static analysis techniques.

4.3.4 Support Checklist

This checklist was created to give support during [PAXSPL](#) execution. It contains items for each activity of our process and may reduce the difficult to conduct the process and to make some decisions as well. 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?
- 1.7. 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 skill 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?

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 process preparation, assembly and execution experience registered?

4.4 Chapter Lessons

In this chapter we described our work main proposal: [PAxSPL](#). [PAxSPL](#) is a process that assembles a feature retrieval process for [SPL](#) re-engineering. We described our process structure including its roles/actors, artifacts, phases, activities and guidelines. The guidelines documentation aims to guides the team applying the process, reducing effort.

5 ASSESSING OUR PROPOSAL

We applied a survey¹ in researchers of the **SPLE** and Re-engineering field to evaluate the impact, relevance and originality of our proposal in the **SPL** re-engineering field. We also conducted a Case Study to collect evidence about **PAXSPL** applicability in comparison to when no process is given. In this chapter we explain the survey structure, Section 5.1, discuss the participants profile, Section 5.1.1, list and discuss the questions and its results, Section 5.1.2, and analyze the survey and its results in Section 5.1.3. In regard to the Case Study (see Section 5.2), we present its planning in Section 5.2.1, the conduction is presented in Section 5.2.2, and its results are discussed in Section 5.2.4.

5.1 Survey

We split the survey in six sections intending to better organize the information, and get better understanding from the participants. The first section is an introduction of the survey, briefly showing the goals of our research and the survey goals as well. The second section of the survey had questions with regard to the participants information such as academic degree, knowledge and experience in the **SPLE** area. The third section had our process documentation, being responsible for giving participants, the information needed to answer the later questions. Fourth and fifth sections contained questions to assess our process. We have used Likert Scale (**LIKERT**, 1932) to give participants options to rate the answers based on their opinion, in which the number one (1) represents a low degree of agreement and number five (5) represents a high degree of agreement with the question. The last sections was created to ask for feedback with regard to the survey, such as clarity of the questions, comments about the survey, and comments about our process.

5.1.1 Participants Profile

To answer our survey we considered researchers working on either the **SPLE** or the re-engineering field. We invited authors of relevant studies in both **SPL** and re-engineering fields. Within the survey we included questions to obtain information about the participants degree, knowledge, skills and experience. The total number of answers was eleven.

With regard to the academic degree, 27% of the participants were undergraduate students, 27% were graduated, 27% were PhD and 19% held a Master degree. This information is important, because it indicates the degree of confidence that we may obtain from the answers.

Regarding the experience with **SPL**, we have asked how many years the participants have been working or researching in the **SPL** engineering field. The answers were

¹ <<http://tiny.cc/surveyluciano>>

Table 8 – Experience applying/performing/monitoring:

Topic	Yes	No
SPL engineering process	55%	45%
Re-engineering process	55%	45%
SPL Re-engineering process	55%	45%

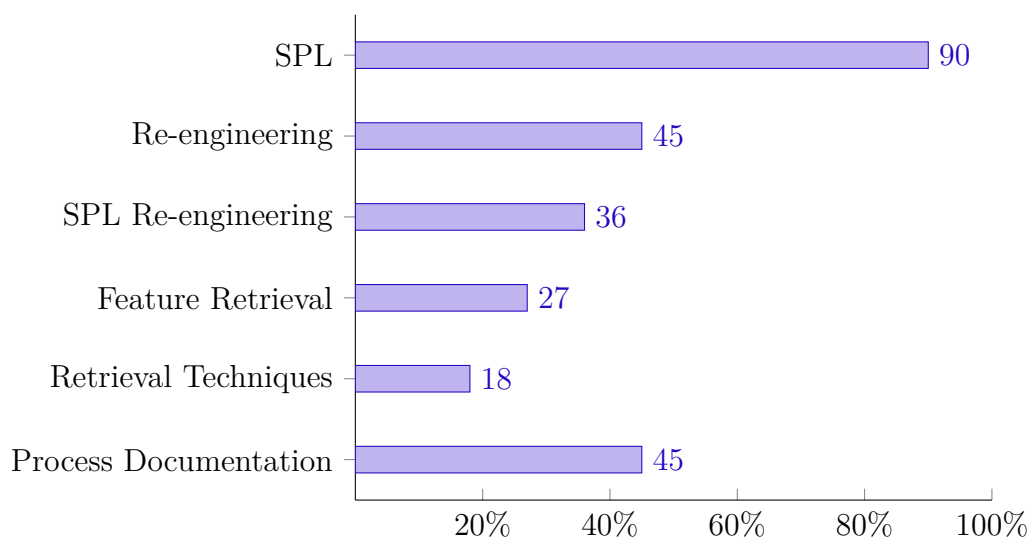
Source: Author.

satisfactory for us, because 64% of the participants have been working or researching in this field for 1 to 5 years. For the rest, 26% have being working or researching for less than 1 year, and 10% for more than 5 years. We believe this information is crucial, because if most of the participants had not enough experience with SPL they would possibly not have enough knowledge to understand and contribute to our survey.

We have also asked if the participants have applied, performed or monitored a SPL engineering process. For this questions, 55% answered *yes* (Table 8), same result of the question regarding if they have applied, performed, or monitored a re-engineering process or a SPL Re-engineering process.

The last part of the participants profile questions was about their technical knowledge related to topics of interest for this work, shown in Figure 11. For the first topic, SPL, 90% of the participants have checked as part of their knowledge. Both Re-engineering process and Process Documentations topic were checked by 45% of the participants. 36% checked the SPL Re-engineering topic, 27% checked Feature Retrieval, and 18% checked Feature Retrieval Techniques. These results indicated that the participants that have answered our survey were able to asses our process with a good level of confidence.

Figure 11 – Participants technical knowledge.

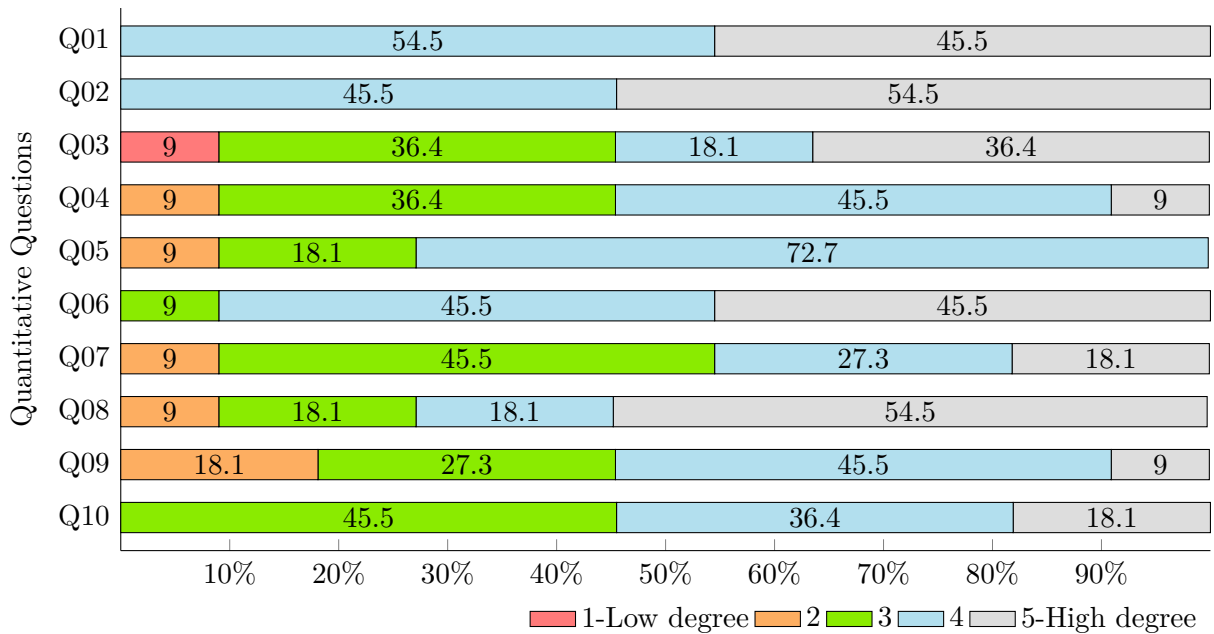


Source: Author.

5.1.2 Results

The results analysis of this section is based on the numbers presented in Figure 12. The first four questions were related to the relevance of our proposal. We asked whether a generic process for performing SPL re-engineering would be a relevant contribution to the field (**Q01**). Most of the participants answered with 4 (54.5%) or 5 (45.5%), as illustrated in Figure 12. These results indicate that a generic process would be relevant.

Figure 12 – Frequency diagram of the quantitative questions.



* Some percentages generate a recurring decimal which do not result in a 100% total.

Source: Author.

The second question was still related with the relevance; however, it was specific for the customization and applicability of the process in different scenarios (**Q02**). We have also obtained similar results as it happened in the first question, where 45.5% of the participants selected 4 and 54.5% answered with 5. For the third question, we asked if our process is a novel contribution to the field (**Q03**). For this question, we obtained mixed results: most of the participants answered with 3 or 5 (36.4% each); The option 4 was selected by a smaller group (18.1%); Lastly, the option 1 was selected by only 9% of the participants and despite being an exception to the answers, it is still a not desired result. We still think, however, that as most of the answers were 3 or higher, our process may be a novel contribution.

The fourth question was related with the impact of our proposed process (**Q04**). This impact would be the importance of our proposal for the SPL re-engineering field. The option 4 (45.5%) and 3 (36.4%) was checked by most of the participants and the options 2 and 5 were both selected by only 9% each. These results also proved to be good

because the majority of the answers indicate that **PxSPL** would have a good impact for the **SPL** re-engineering.

For the second part of the questions, showed in Figure 12, we asked a question related with the flexibility of our process to collect information about the team, assign roles and define tasks (**Q05**). The goal of this question is to evaluate the flexibility level of our process exclusively for those topics. We obtained good results again, as the great majority of the participants chose the option 4 (72.7%), with a few choosing 3 (18.1%) and only 9% choosing 2. This implies that our process provides flexibility regarding the team management topic.

For the next two questions, we have asked exclusively about the artifacts documented within our process. The first question was regarding the relevance of a process that provides flexibility for the artifacts used (**Q06**). Both the options 5 and 4 were selected by most of the answers (45.5% each) and 3 was selected by only 9% of the participants. The second question was whether the participants thought that our process is a novel contribution in regards only to its flexibility related with artifacts (**Q07**). We have obtained another mixed set of answers, as most of the participants chose 3 (45.5%), the option 4 was selected by 27.3%, option 5 was selected by 18.1% and option 2 was selected by 9%. As it happened with the flexibility of the team management topic, we conclude that, although our process is flexible in the artifacts topic, it could be improved.

For the next two questions, we asked if, considering the flexibility related with techniques for feature retrieval, a **SPL** re-engineering process is a relevant contribution (**Q08**). Once again we got mixed answers, where most of the participants, chose the option 5 (54.5%). The options 3 and 4, got the same amount of answers (18.1% each) and 9% of the participants selected option 2. We have also asked if, regarding the flexibility related with techniques for feature retrieval, our process is a novel contribution to the **SPL** re-engineering field (**Q09**). For this question, most of the participants have answered with 4 (45.5%), following by a smaller group answering with 3 (27.3%), 18.1% have answered with 2 and also 9% of the participants answering with 5. Once again we obtained mixed answers; though most of them were positive, we think that some improvements should be made to make our process more flexible in those topics.

The last question was related with the applicability of our process (**Q10**). For this question, a great majority of the participants have answered with 3 (45.5%), following by participants that selected 4 (36.4%) and 5 (18.1%). This indicates that for most of the participants, our process is considerably applicable.

5.1.3 Results Analysis

In the last section of the survey we have placed two open questions to give participants the opportunity to leave comments and suggestions for our process. Most of the suggestions were related to expand the flexibility of our proposal. Some participants in-

indicated the need for improving some aspects of the documentation to obtain more clarity about how flexible the process really is.

Another suggestion was related to the process been specific in some points such as artifacts used by some retrieval techniques. Some participants suggested to define this techniques in a generic way and provide supporting tools and methods as a recommendation for executing each technique.

By analyzing the results of the questions and the comments and suggestions as well, we have concluded that our process is on the right direction towards our main goal. We could noticed that there is still some lack of flexibility in some points. Despite these problems, the results indicated the impact, relevance and originality of our proposal.

For the first topic, we could see with the results of the questions that a generic process that provides flexibility for [SPL](#) Re-engineering would make impact in the field. This indicates that the motivation for creating our process makes sense and we have been doing a significant research in the area.

With regard to the relevance of our proposal, the results showed that our proposal is being developed towards the direction we intended. We concluded, however, that our proposal may not be a novel contribution as we toughed. However, we do not think this is a problem, because our process will be improved and expanded in future works, what may turn it into an unique proposal.

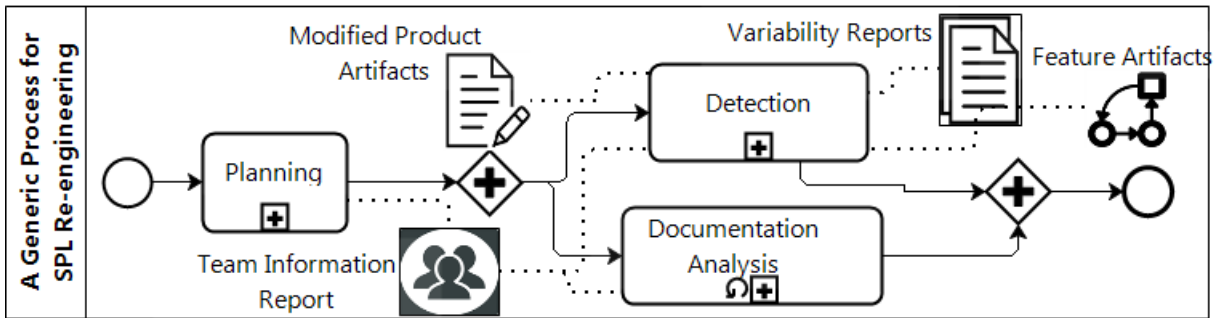
5.1.4 Process Improvements considering the Survey Results

The process presented in Chapter 4 is the result of the changes made after the survey application, thus it is the latest version of our process. We think, however, that it is important to present the first version of our process pointing the main differences between each version.

The first version of our process had only three activities in the main flow, as Figure 13 shows. There are some artifacts showed in the figure (Team Information Report, Modified Product Artifacts, Variability Reports, Feature Artifacts) that are still part of our process (see Section 4.1.3), and some activities (Planning, Documentation Analysis) as well. The first version, however, did not include a feature model as an output artifact. In the first version we did not intend to generate a feature model, only retrieve the features, without further analysis.

Another difference is the artifact called Assembled Process (see Section 4.1.3.3), which was not present in the first version of our proposal. The reason to create this process is related with some comments of the survey that pointed to the need of formalizing the techniques selection. In the first version, we did not have an artifact to formalize the selected techniques. This selection was made during an activity called Feature Search, as illustrated in Figure 14. With the introduction of the Assembled Process, we provide to the team a way to assemble the selected techniques into a process. We still, however,

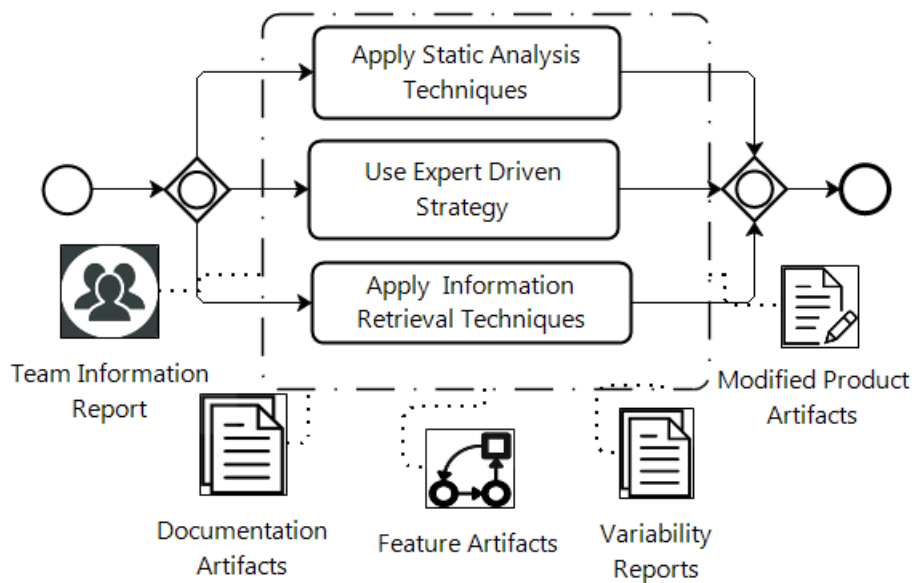
Figure 13 – First version of our process.



Source: Author.

maintain flexibly because they may assemble the techniques as they prefer.

Figure 14 – First version of our process (Feature Search Activity).



Source: Author.

Another difference is the introduction of new sections in the guidelines documentation (see Section 4.3), such as the feature model for retrieval techniques, and the tools recommendation for each technique.

5.2 Case Study

5.2.1 Rationale and Preparation

To further evaluate PAXSPL we conducted a case study in a real development environment. To the best of our knowledge there is not a process that gives support to feature retrieval according to a specific scenario. Thus we decided to collect evidence of

PAxSPL when applied in an organization that does not have a re-engineering process. We also wanted to collect evidence about the reuse capability provided by **P**AxSPL when performed in a different scenario. As an exploratory evaluation, this Case Study was focused in qualitative results rather than quantitative. Based on this we formulated our objectives, **R**Qs and data collection mechanisms.

5.2.1.1 Objectives and Research Questions

The main objective of the Case Study is to evaluate how **P**AxSPL may help a team with little or no experience in SPL re-engineering to perform feature retrieval in a set of system variants. We wanted to collect exploratory evidence about the most relevant **P**AxSPL artifacts, the quality of these artifacts, effort to apply **P**AxSPL, and **P**AxSPL reuse capability when re-applied in a different scenario.

Based on the objectives we formulated the following **R**Qs:

- RQ1.** What is the impact/relevance of the artifacts generated by **P**AxSPL execution in terms of re-engineering?
- RQ2.** What is the quality of the generated artifacts?
- RQ3.** What is the effort needed to perform each phase of **P**AxSPL?
- RQ4.** What is the reuse capability of **P**AxSPL?

The impact/relevance of the artifacts generated during **P**AxSPL execution is measured in terms of the re-engineering process (**RQ1**). We want to measure how many times and in which activities each artifact was used. We also asked the participants about their opinion of each artifact relevance/impact when choosing techniques for feature retrieval.

In regard to the quality of the generated artifacts (**RQ2**), we created quality metrics (see Section 5.2.1.5). We used these metrics to measure how much the generated artifacts are satisfactory for the process execution in terms of re-engineering, giving us evidence to answer **RQ3** and **RQ4** without have to consider quality differences.

The effort comparison of each phase (**RQ3**) is in terms of: time spent to learn concepts needed to perform feature retrieval and analyze product documentation (**Prepare** phase); time spent to select techniques and assemble them before retrieve the features (**Assemble** phase); and time spent to perform the retrieval techniques to generate the feature artifacts (**Execute** phase); In addition, we also considered the complexity of each activity in regard to the participants opinion.

To gather information for **RQ4**, we considered which artifacts could be reused without any modification. We also considered the participants opinion about artifacts reusability. To maintain the quality level for both phases, we considered the artifacts quality scores from both phases.

5.2.1.2 Context and Units of Analysis

The organization where the case study was applied is a startup called Porthal². Porthal possesses a web-based enterprise resource planning system called ICode. ICode can be used for managing registration of customers, suppliers, tax accounts, finances, schedules, products, sales and electronic invoice generation. ICode was developed using a modular architecture, composed of six different modules:

- *Cadastro*³: this module contains all features related with registers of clients, products, suppliers, customers, and so on;
- *Tributário*: in this module features related to accountant are implemented, such as taxation related functionalities;
- *Financeiro*: module composed by features related with finances, such as bank and cards account information;
- *NFe*: module in which all features related with electronic invoice are implemented, such as generation and validation of electronic invoice;
- *Agenda*: module composed by features related with schedule plans and appointments;
- *Pedido*: module where sales requests and generation of related documents are performed.

ICode was developed using JAVA, using the JAVA EE API, and Java Server Faces (JSF). The current version of ICode is composed of more than two hundred JAVA class and has a database with more than sixty tables.

Porthal attends more than fifty clients and distributes different configurations of ICode (products) according to their clients' needs. For instance, a product may have three modules (*Cadastro*, *Financeiro* and *Agenda*). However, this product configuration is performed in a manual way. No systematic solution, such as SPL, is used. In addition, there is no artifacts created to model the variability among the possible configurations. These characteristics make ICode have potential to become SPL, being a good choice for this case study. However, Porthal had not applied or even selected a re-engineering process. Therefore we decided to observe how the software engineers would perform the feature retrieval using PAXSPL.

With regard to the participants, they were software engineers from the organization where the case study was applied. They have experience working as software developers, and were grouped in a team.

² <<http://porthal.com.br/>>

³ When describing the system modules and features, we used the system's original language, which is Portuguese.

5.2.1.3 Case Study Protocol

The case study protocol was divided in two phases. During **Phase 1**, **PAXSPL** was performed in five different products of ICode. These five products were analyzed and the feature retrieval process was executed. The five first products were composed by a different combination of four modules:

- **Product 1:** *Cadastro* and *Tributário*;
- **Product 2:** *Cadastro* and *Financeiro*;
- **Product 3:** *Cadastro*, *Tributário*, *Financeiro* and *NFe*;
- **Product 4:** *Cadastro*, *Tributário* and *NFe*;
- **Product 5:** *Cadastro*, *Tributário* and *Financeiro*;

In **Phase 2**, we re-applied **PAXSPL** adding five more products, thus the participants should use ten different products of ICode as input for the feature retrieval. The five products added in **Phase 1** were composed by a different combination of six modules:

- **Product 6:** *Cadastro*, *Tributário*, *NFe* and *Agenda*;
- **Product 7:** *Cadastro*, *Financeiro* and *Pedido*;
- **Product 8:** *Cadastro*, *Tributário*, *Financeiro*, *NFe* and *Agenda*;
- **Product 9:** *Cadastro*, *Tributário*, *Financeiro*, *NFe*, *Agenda* and *Pedido*;
- **Product 10:** *Cadastro*, *Tributário*, *Financeiro* and *Pedido*;

The documentation generated during the first execution was used, allowing to reuse the generated artifacts in a different set of system variants.

We did not set any time limit for the participants, and the case study was performed during the organization working hours (8 hours a day). We still, however, registered the time spent for each activity of **PAXSPL**.

5.2.1.4 Data Collection

To increase the precision and strengthen the validity of our case study we performed the data collection taking multiple perspectives, thus allowing triangulation (RUNE-SON et al., 2012). Therefore we collected data using different sources: observing and documenting information in real time, applying different surveys⁴, and analyzing **PAXSPL** generated artifacts. In regard to information documented while observing, we monitored

⁴ The surveys are available at <<https://goo.gl/uFPQXH>>

and registered the time spent in each activity of **PAXSPL**. We also kept track of each artifact used during each activity, registering how often and in how many activities an artifact was used. During **Phase 2** of the case study we compared these information to answer **RQ4**.

We applied three surveys in different moments of the case study. Survey 1 was applied before **Phase 1** aiming to collect information about the participants. Survey 2 was applied before **Phase 2** to collect the information gathered during **Phase 1**, aiming to answer **RQ1**, **RQ2** and **RQ3**. Lastly, Survey 3 was applied after **Phase 2** to collect information related to **RQ4**. The artifacts were analyzed only at the end of the whole execution, aiming to measure their quality.

5.2.1.5 Quality Metrics

To measure the quality of the artifacts generated during **PAXSPL** we created a Quality Metrics (**QM**). For each **QM** we have three possible scores: Total (**T**) which scores 1, Partial (**P**) which scores 0.5, and Nothing (**N**) which scores 0 (zero). The **QMs** are described as follows:

QM1. Are the **PAXSPL** mandatory artifacts well documented following their templates?

Evaluation - **T**: all mandatory artifacts are documented and follow their templates when provided; **P**: a few mandatory artifacts were not documented or did not follow their templates; **N**: most of the mandatory artifacts were not documented;

QM2. Are all the team members roles and tasks documented in the team information report?

Evaluation - **T**: all team members mandatory information is registered in the team information report; **P**: some mandatory information is missing from the team information report; **N**: the team information report was not documented at all;

QM3. Were all needed product artifacts analyzed/documented during the documentation analysis?

Evaluation - **T**: every needed product artifact of the documentation analysis activity was analyzed/documented; **P**: not all product artifacts were not analyzed/documented; **N**: only a few needed product artifacts were analyzed/documented;

QM4. Were the decision and reasons to select or not select a technique for feature retrieval documented?

Evaluation - **T**: the decisions and reasons to select or not every candidate technique were documented; **P**: most of decisions and reasons to select or not a technique were documented; **N**: only a few candidate techniques had its reasons for select it or not documented;

QM5. Was the assembled process documented following a process structure?

Evaluation - **T**: the assembled process was documented following a structure showing its activities, roles, work-flow and artifacts; **P**: not all details of the assembled process were documented; **N**: none document was created for the assembled process.

QM6. Were all the retrieval techniques output artifacts documented?

Evaluation - **T**: every retrieval technique output artifact was documented; **P**: most retrieval techniques output artifacts were documented; **N**: only a few retrieval techniques output artifacts were documented.

QM7. Were all feature entry points from the retrieved features documented?

Evaluation - **T**: all feature entry points from the retrieved features were documented; **P**: most feature entry points from the retrieved features were documented; **N**: only a few feature entry points from the retrieved features were documented.

QM8. Were all features from the system variants retrieved?

Evaluation - **T**: all features from the system variants were retrieved and documented; **P**: most features from the system variants were retrieved and documented; **N**: only a few features from the system variants were retrieved and documented.

QM9. Was a feature diagram created to represent system variants as **SPL**?

Evaluation - **T**: a feature diagram was created following a feature model notation; **P**: a feature diagram was created, however none feature model notation or rules was followed; **N**: a feature diagram was not created.

QM10. Was the process execution experience documented?

Evaluation - **T**: the process execution experience and all decision points were documented following a template; **P**: the process execution experience was documented, however no template was used; **N**: the process execution experience was not documented at all.

We used these ten **QMs** to measure and compare the quality of the generated artifacts during the two phases of our case study (**RQ2**).

5.2.2 Conduction

Before the case study execution, we did not give any kind of training or introduction to basic **SPL** or re-engineering concepts to the participants. At the beginning of **Phase 1**, the participants received **PAsPL** documentation and the five first products of **ICode**. The **ICode** artifacts were composed by source code, requirements list and some architecture diagrams.

During **PAXSPL** execution, we observed and documented the information that would be used to answer the **RQs**.

At the beginning of **Phase 1**, we gave to the participants five more products of **ICode**, composed by the same types of artifacts. They also could use the documentation generated during **Phase 1**. We collected the same kind of information, allowing us to perform some comparison between both phases.

5.2.3 Collected Data

The data collected to answer the **RQs** can be categorized in four different categories: time spent, quality of artifacts, use and reuse of artifacts, and the answers from the surveys. This data is detailed in the following sub-sections.

5.2.3.1 Time Spent

The case study was executed during two weeks. Table 9 details the information about the time spent in each phase and activity of **PAXSPL** during both phases of the case study. In total, **Phase 1** was performed in thirteen and a half hours, while for **Phase 2**, only four and a half hours were needed. For both execution phases, the most time consuming phase of **PAXSPL** was **Execute** (eight hours for **Phase 1** and three and a half hours for **Phase 2**), mainly because the **Execute Assembled Process** activity, which was reduced in three hours. Two activities were reduced by one hour during **Phase 2**: **Perform Documentation Analysis** was performed and one and a half hour during **Phase 1** and half an hour during **Phase 2**, while **Document Feature Artifacts** reduced from two hours to one. Another important information is the activities where no time was spent during **Phase 2** of the case study. During these activities (**Collect Team Information**, **Assign Roles** and **Select Techniques**), the artifacts from **Phase 1** were reused, justifying the time saved.

Table 9 – Time spent in each activity of **PAXSPL** during both phases of case study.

Activity/Time Spent (Hours)										
	Prepare			Assemble			Execute			Total
	Collect Team Info.	Assign Roles	Perform Doc. Analysis	Select Tec.	Assemble Tec.	Assign Tasks	Exec. Assembled Process	Doc. Feature Artifacts	Doc. Process Exp.	
P1	0.5	0.5	1.5	1.5	1	0.5	5	2	1	13.5
P2	0	0	0.5	0	0.2	0.3	2	1	0.5	4.5

P1 - Phase 1; P2 - Phase 2.

Source: Author.

5.2.3.2 Quality of Generated Artifacts

During execution of both phases of the case study more than 30 artifacts from **PAXSPL** were generated. This however, does not give evidence that the process was fol-

lowed as required. In addition, the generation of an artifact does not indicated its usability and quality. Thus, at the end of the case study we analyzed the artifacts and applied ten QMs (see Section 5.2.1.5) to give evidence that the artifacts were generated according to the process guidelines. The score obtained from **Phase 1** of the case study was 9.5 while from Phase 2 was 10, as showed in Table 10. The only difference between both phases was in QM6, where some feature artifacts were not documented during **Phase 1**.

Table 10 – Quality of PAXSPL artifacts generated during both phases of case study.

	QMs/Scores										
	1	2	3	4	5	6	7	8	9	10	Total Score
P1	T	T	T	T	T	P	T	T	T	T	9.5
P2	T	T	T	T	T	T	T	T	T	T	10

P1 - Phase 1; P2 - Phase 2.
T - Total Score; P - Partial Score.

Source: Author.

Among the generated artifacts⁵, at least one was generated during each activity of PAXSPL. For the first two activities, **Collect Team Information** and **Assign Roles** we have the Team Information Report. As is showed in Figure 15, the report was documented following its template (see Appendix A), indicating that the process description was followed for those activities. The artifact described members (for ethical reasons we asked the participants to do not use their names), their roles, experience working with SPL and knowledge with feature retrieval techniques. The generated artifact in both phases of the case study was the same, because the team members did not change.

Another important artifact is the retrieval techniques report, alongside the configured feature model of retrieval techniques. Both artifacts are generated during **Select Techniques** sub-process. For the first artifact, the retrieval techniques report, showed in Figure 16, we also noticed that the template (see Appendix C) was followed and all candidate retrieval techniques were documented describing the reasons for their selection or not. In this case, the candidate techniques were FCA, Clustering and Expert-Driven Extraction which were selected, and LSI which was not.

Besides these artifacts importance during PAXSPL execution, the feature retrieval artifacts generated by the application of retrieval techniques have direct relation with the final output of PAXSPL, which is a feature model.

For each retrieval technique applied (with exception of expert-driven extraction) at least one artifact was generated. For instance, as showed in Figure 17, a concept lattice generated during **Phase 2** of the case study. This lattice presents an ordered hierarchy of the features extracted from ten different products of ICode. In this case, the more close

⁵ All artifacts available at <<https://github.com/HestiaProject/PAXSPL/tree/master/process/case%20study/Generated%20Artifacts>>

Figure 15 – Team information report generated during case study.

Team Information Report

Created by: Team Member 1

Date: 11/04/18

Member:	<i>Team Member 1</i>
Email:	
Roles in company:	<i>Software Engineer</i>
Roles during product development:	<i>Software Engineer, Developer</i>
Experience working with SPL	<i>Participant in a quasi-experiment about SPL re-engineering</i>
Knowledge about retrieval techniques	<i>Applied Formal Concept Analysis during a quasi-experiment.</i>
Obs:	
Role During re-engineering:	<i>Feature Retriever, Developer, Analyst</i>

Member:	<i>Team Member 2</i>
Email:	
Roles in company:	<i>CEO and Software Engineer</i>
Roles during product development:	<i>Developer, Product Owner, Software Engineer</i>
Experience working with SPL	<i>Current Working with SPL in his Master Degree</i>
Knowledge about retrieval techniques	<i>None</i>
Obs:	<i>Domain Specialist, worked with the system for more than 5 years.</i>
Role during re-engineering:	<i>Domain Engineer, Feature Tester</i>

Source: Author.

to the top the features are showed, the more products of ICode they are part of. For instance, features *Veiculo*, *Cidade*, *Estado*, *Pais*, *Vendedor*, *Unid Medida*, *Transportadora*, *Fornecedor*, *Produto* and *Cliente* are shared among all ten products. Features such as *Tarefas*, *Historico* and *Geral* are shared only by v6, v8 and v9, while *Status Pedido*, *Gerar NFe* and *Realizar Pedido* are shared only by v7, v10 and v9.

The concept lattice, however, is not a mandatory artifacts and was generated because the participants selected **FCA** as an extraction technique. By using the information analyzed from the lattice, the participants were able to create clusters, validate them with the Domain Specialist, and generate the feature model.

The final output of **PAXSPL**, which is the feature model of the extracted features, was did not suffer much change during both phases of the case study. The main difference between **Phase 1** (Figure 18) and **Phase 2** (Figure 19), was the addition of the features (*Pedido*, *Agenda* and their sub-features) extracted from the new products included in **Phase 2**. Considering the similar features, we have *Cadastro* and its sub-features as mandatory, because they were present in all products. *NFe* and *Pedido* are part of an *or*-alternative group, because at least one of them must be present when the optional feature *Documento* is part of the product. We have the abstract feature *Operacoes*, which is mandatory and compose of an *or*-alternative group of the sub-features *Tributario* and *Financeiro*. There is also the optional feature *Agenda* (only in feature model from **Phase**

Figure 16 – Retrieval techniques report generated during case study.

Candidate Technique:	<i>Formal Concept Analysis</i>
Related Techniques:	<i>Clustering, LSI</i>
Selection Result	<i>Selected</i>
Reasons	<i>Prior experience of team members, practical examples are similar to current scenario, artifact inputs match.</i>
Obs:	

Candidate Technique:	<i>Clustering</i>
Related Techniques:	<i>Formal Concept Analysis</i>
Selection Result	<i>Selected</i>
Reasons	<i>Practical scenarios are similar to current. Also team members have little experience using it.</i>
Obs:	

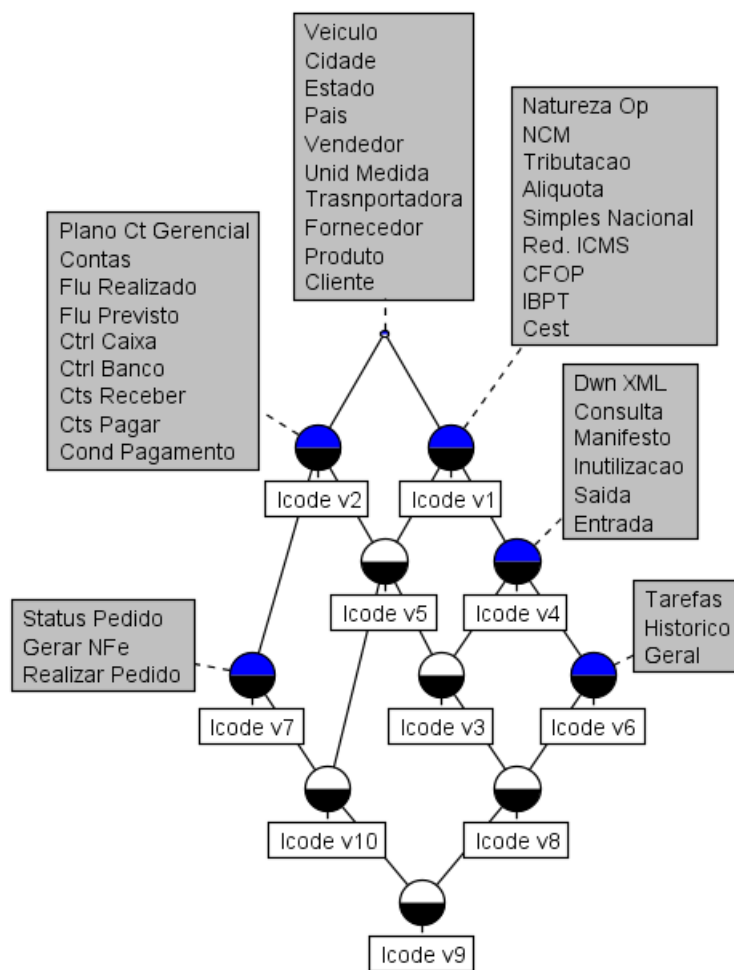
Candidate Technique:	<i>Latent Semantic Indexing</i>
Related Techniques:	<i>Clustering, Formal Concept Analysis</i>
Selection Result	<i>Not Selected</i>
Reasons	<i>Practical scenarios are different from current. Also team members have no experience using it.</i>
Obs:	

Candidate Technique:	<i>Expert Driven Extraction</i>
Related Techniques:	
Selection Result	<i>Selected</i>
Reasons	<i>Domain Specialist possesses huge knowledge about products.</i>
Obs:	

Source: Author.

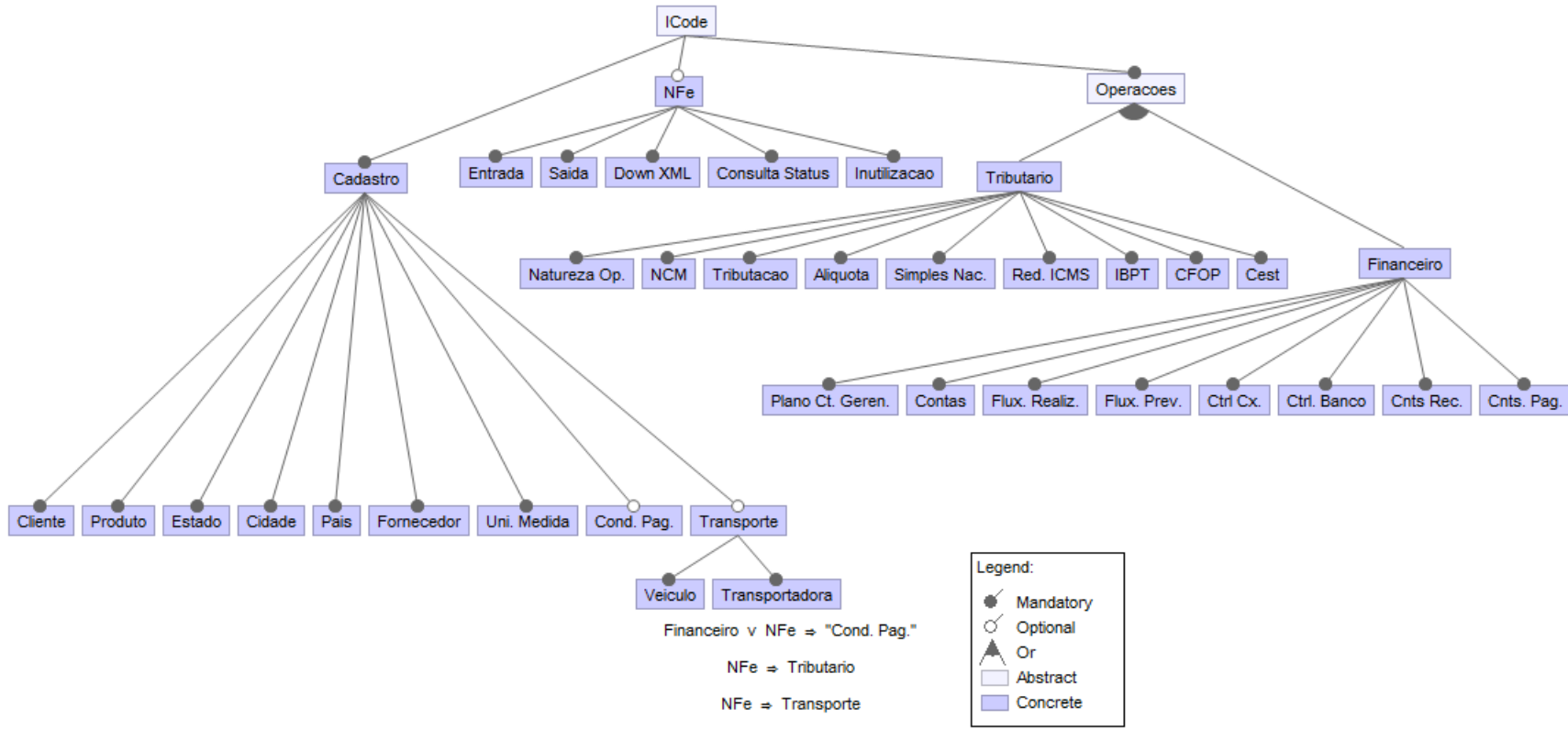
2). Also, some restriction were documented in the feature model: if the features *Financeiro* or *NFe* are part of a product, than the feature *Cond Pag* must also be a part of that product. The feature *NFe* requires the features *Tributario* and *Transporte*. Lastly, the feature *Gerar NFe* requires *NFe*.

Figure 17 – Concept lattice of features extracted from ten different products of ICode.



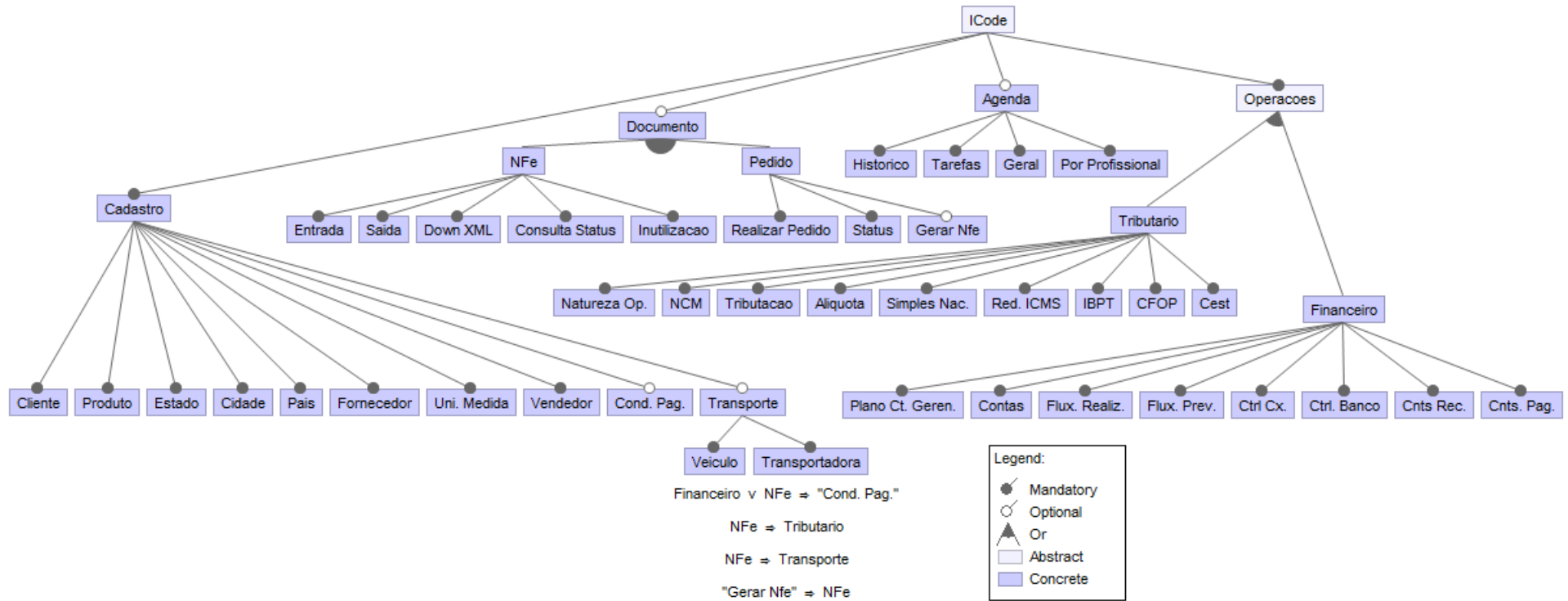
Source: Author.

Figure 18 – ICode feature model from Phase 1.



Source: Author.

Figure 19 – ICode feature model from Phase 2.



Source: Author.

5.2.3.3 Use of Generated Artifacts

We also collected the number of uses of each artifact during each activity of **PxSPL**. As presented in Table 12, the artifacts most used, used in six different activities, were Team Information report, Assembled Process, Process Documentation and **PxSPL** Guidelines. The artifact less used was Development Information Report. Despite these results, this does not give enough evidence of the impact/relevance level of the artifacts, we still had to consider the surveys answers (see Section 5.2.3.4).

Another type of data collected was the reusability of the generated artifacts. We documented, as showed in Table 11, the use condition of the artifacts during **Phase 2** of the case study. Only one artifact from **Phase 1** was not used (Development Information Report). Four artifacts were reused with no need to perform modifications. Most of the artifacts generated during **Phase 2** were updated, needing some modification during **Phase 2**.

Table 11 – Artifacts reused during Phase 2.

Artifact	Condition of use during Phase 2		
	Used without modification	Updated	Not Used
Team Information Report	✓		
Artifacts Type Specification	✓		
Development Information Report			✓
Domain Glossary		✓	
Domain Constraints List		✓	
Requirements Specification		✓	
Retrieval Techniques Report	✓		
FM of Retrieval Tec. Configured	✓		
Assembled Process		✓	
Process Documentation		✓	
Feature Report		✓	
Feature Model		✓	

Source: Author.

Table 12 – Use of artifacts for each activity of PAXSPL.

Activities/Use										
Artifact	Collect Team Information	Assign Roles	Perform Documentation Analysis	Select Techniques	Assemble Techniques	Assign Tasks	Exec. Assembled Process	Document Feature Artifacts	Document Process Experience	Total
Team Information Report	P1, P2	P1, P2		P1		P1				6
Artifacts Type Specification			P1,P2	P1						3
Development Information Report			P1	P1						2
Domain Glossary			P1,P2	P1						3
Domain Constraints List			P1,P2	P1						3
Requirements Specification			P1,P2	P1			P1,P2			5
Retrieval Techniques Report			P1,P2	P1	P1					4
FM of Retrieval Tec. Configured			P1,P2	P1,P2						4
Assembled Process						P1,P2	P1,P2		P1,P2	6
Process Documentation						P1,P2	P1,P2		P1,P2	6
Feature Report							P1,P2	P1,P2		4
Feature Model							P1,P2	P1,P2		4
PAXSPL Guidelines			P1,P2	P1,P2			P1,P2			6

P1 - Used during Phase 1; P2 - Used during Phase 2.

Source: Author.

5.2.3.4 Surveys Answers

Survey 1 was designed to collect information about the participants and their knowledge about **SPL**, thus its answers were almost not used for the triangulation analysis of artifacts.

Otherwise, Survey 2 had a higher contribution to the analysis of the collected data. The first questions we highlight are related with the importance/relevance of **PxSPL** artifacts. We asked the participants to list the most important artifacts in terms of re-engineering. The cited artifacts were: team information report, requirements specification, assembled process and **PxSPL** guidelines.

We also asked them to give a score (1 to 5) about the relevance of each artifact. These same questions were also part of Survey 3 and were used to answer **RQ1**. We calculate the average of each survey and then the average considering both surveys. As presented in Table 13, the most relevant artifacts in terms of re-engineering according to the survey results are the requirements specification, the assembled process, the domain constraints list, the retrieval techniques report, and the **PxSPL** guidelines. The less relevant artifacts are: artifacts type specification, development information report, domain glossary and feature model of retrieval techniques configured. Other artifacts can be considering as mid-level relevance: team information report, process documentation and feature report.

Table 13 – Relevance of artifacts according to survey results.

Artifact	Relevance Level		
	Survey 2 average	Survey 3 average	Total average
Team Information Report	3	3	3
Artifacts Type Specification	2	2	2
Development Information Report	3	1	2
Domain Glossary	2	3	2.5
Domain Constraints List	4	4	4
Requirements Specification	5	5	5
Retrieval Techniques Report	3	5	4
FM of Retrieval Tec. Configured	2	2	2
Assembled Process	5	5	5
Process Documentation	4	3	3.5
Feature Report	3	4	3.5
PxSPL Guidelines	5	3	4

1 - Very low relevance; 5 - Very high relevance.

Source: Author.

Another important question present in both surveys (2 and 3) was related to the complexity to perform each **PxSPL** activity. The participants should give a score (1 to 5) according to the complexity they had to perform each activity. Table 14 shows that according to the answers, the most complex activities are: **Select Techniques** and **Assemble Techniques**. Activities such as **Collect Team Information**, **Assign Roles**, **Assign Tasks** and **Document Process Experience** have a lower level of complexity. In

in addition Perform Documentation Analysis, Execute Assembled Process and Document Feature Artifacts are categorized as mid-level of complexity to be performed. We plan to use these information to help us answer RQ3.

Table 14 – Complexity level of each PAXSPL activity according to survey answers.

Complexity Level			
Activity	Survey 2 average	Survey 3 average	Total average
Collect Team Information	1	1	1
Assign Roles	2	1	1.5
Perform Doc. Analysis	4	3	3.5
Select Techniques	5	5	5
Assemble Techniques	5	4	4.5
Assign Tasks	2	2	2
Execute Assembled Process	4	3	3.5
Document Feature Artifacts	3	2	2.5
Document Process Exp.	2	1	1.5

1 - Very low complexity; 5 - Very high complexity.

Source: Author.

Survey 3 focused on questions about the reuse capability of PAXSPL, aiming to give us information to help answer RQ4. We asked which PAXSPL artifacts from Phase 1 of the case study provided the most level of reuse. According to the participants these artifacts are: team information report, retrieval techniques report and assembled process. Table 15 summarizes the answers according to the level of relevance of each artifact generated during the Phase 1 of the case study.

Table 15 – Reusability level of PAXSPL generated artifacts according to survey 3 results.

Reusability Level	
Artifact	Average
Team Information Report	5
Artifacts Type Specification	5
Development Information Report	5
Domain Glossary	3
Domain Constraints List	2.5
Requirements Specification	2
Retrieval Techniques Report	5
FM of Retrieval Tec. Configured	5
Assembled Process	4.5
Process Documentation	3.5
Feature Report	2.5
Feature Model	4

1 - Low reusability; 5 - High reusability.

Source: Author.

The artifacts with higher reusability are: team information report, artifacts type specification, development information, retrieval techniques report, feature model of retrieval techniques configured, assembled process and feature model. The less reusable artifacts are: domain glossary, domain constraints list, requirements specification and feature report.

5.2.4 Results and Discussion

To achieve the information required to answer the RQs, we perform a cross-analysis among the data collected from different sources during the case study. Thus, we analyzed and merged data collected in real time, survey answers, and generated artifacts analysis. As this case study was an exploratory study, the main goals of the RQs are related to find evidence about opportunities to improve PAXSPL. The cross-examination of those data is detailed in the following sections.

5.2.4.1 RQ1. What is the impact/relevance level of the artifacts generated by PAXSPL execution in terms of re-engineering?

The objective of this RQ is to collect evidence about which PAXSPL artifacts are more relevant during the execution of the process. The reason to calculate this relevance is to give focus on this artifacts for future improvements. We also wanted to identify which artifacts have more impact during **Select Techniques** sub-process. To answer this first question, we perform a cross-analysis in data presented in Table 12 and Table 13, alongside additional information gathered during the survey answers and artifacts analysis.

Considering Table 12, which presents the number of uses for each activity of PAXSPL, we could analyzed how useful an artifact may be for each part of the process. This information, however, is not enough to indicate the relevance of the artifacts. Hence, by merging this information with Table 13, which presents the relevance of each artifact according to participants opinion, we can identify if the use of each artifact is somehow related with its relevance. In addition, by considering the artifacts used during **Select Techniques**, we could indicate the impact level of those artifacts while selecting the retrieval techniques.

To calculate the relevance of the artifacts we applied a score count on them. We considered how many times the artifact was used for each phase of PAXSPL (**Prepare**, **Assemble** and **Execute**), we also considered the total number of uses. The total number of uses from the artifacts presented in Table 12 had a range from two to six uses. Thus, the numerical range was composed by five numbers which were used when calculating the score. Therefore, if an artifact was used two times, we added 1 point to the relevance score, if it was used six times, we added 5 points. Considering PAXSPL phases, however, the numerical range was between zero and 4. Thus, when calculating the relevance for

each phase individually, if an artifact was used four times in a specific phase it would receive a score of 5, if it was not used would receive 1 point.

We merged these points with the relevance level given by the participants in the survey (1 to 5), giving the possibility for each artifact get a total score of 10. In addition, we considered if the artifact was listed as most important, in a different survey question. Thus, the final score of each artifact could reach a maximum of 12 points. As presented in Table 16, the artifacts which obtained the higher scores are: team information report, requirements specification, assembled process and PAXSPL guidelines.

Table 16 – Relevance score of each artifact.

Artifact	Number of Uses				Survey Score	Listed as most relevant	Score (P/A/E/T)
	P	A	E	T			
Team Information Report	4	2	0	6	3	✓	10/8/6/10
Artifacts Type Specification	2	1	0	3	2		5/4/2/4
Development Information Report	1	1	0	2	2		4/4/2/3
Domain Glossary	2	1	0	3	2.5		5.5/4.5/3.5/4.5
Domain Constraints List	2	1	0	3	4		7/6/4/6
Requirements Specification	2	1	2	5	5	✓	10/9/10/11
Retrieval Techniques Report	2	2	0	4	4		7/7/4/7
FM of Retrieval Tec. Configured	2	2	0	4	2		5/5/2/5
Assembled Process	0	2	4	6	5	✓	7/10/12/12
Process Doc.	0	2	4	6	3.5		3.5/6.5/8.5/8.5
Feature Report	0	0	4	4	3.5		3.5/3.5/8.5/6.5
PAXSPL Guidelines	2	2	2	6	4	✓	9/9/9/11

P - Prepare; A - Assemble; E - Execute; T - Total;

Source: Author.

To categorize the artifacts, considering the re-engineering, we created four levels of relevance, based on the score:

1. **Not relevant:** artifact was not used.
2. **Useful:** artifact was used and scored 4 relevance points or less.
3. **Helpful:** artifact was used and scored between more than 4 and a maximum of 8 relevance points.
4. **Relevant:** artifact was used and scored more than 8 relevance points.

We categorized the artifacts considering **PAxSPL** phases individually (*e.g.*, development information report was not used during **Execute**) and in general. We summarize these results in Table 17, showing the relevance level of each artifact for **Prepare**, **Assemble** and **Execute** phases, and overall relevance as well.

The information in Table 17 indicates that artifacts such as team information report, requirements specification and **PAxSPL** guidelines are more relevant during the preparation before assembling the process. In addition, requirements specification, assembled process and **PAxSPL** guidelines are more important during the assembly of the retrieval process. For the execution of the feature retrieval, artifacts such as the requirements specification, assembled process, process documentation, feature report and **PAxSPL** guidelines are more relevant. Considering the overall relevance, the more important artifacts are: team information report, requirements specification, assembled process, process documentation and the **PAxSPL** guidelines.

Table 17 – Relevance level of each artifact during **PAxSPL** execution.

Relevance Level				
Artifact	Prepare	Assemble	Execute	Overall
Team Information Report	Relevant	Helpful	Not Relevant	Relevant
Artifacts Type Specification	Helpful	Useful	Not Relevant	Useful
Development Information Report	Useful	Useful	Not Relevant	Useful
Domain Glossary	Helpful	Helpful	Not Relevant	Helpful
Domain Constraints List	Helpful	Helpful	Not Relevant	Helpful
Requirements Specification	Relevant	Relevant	Relevant	Relevant
Retrieval Techniques Report	Helpful	Helpful	Not Relevant	Helpful
FM of Retrieval Tec. Configured	Helpful	Helpful	Not Relevant	Helpful
Assembled Process	Not Relevant	Relevant	Relevant	Relevant
Process Documentation	Not Relevant	Helpful	Relevant	Relevant
Feature Report	Not Relevant	Not Relevant	Relevant	Helpful
PAxSPL Guidelines	Relevant	Relevant	Relevant	Relevant

Source: Author.

We also analyzed the artifacts used only during the **Select Techniques** sub-process to categorize their impact level when selecting techniques. For this analysis, we found important evidence in the retrieval techniques report, where the reasons for selecting a technique are documented. Considering this information, we tabulated the impact level of each artifact when deciding which technique to use for feature retrieval. As showed in Table 18, team information report and requirements specification have both a high level of impact. Domain glossary and domain constraints analysis have a medium level of impact when selecting techniques. Lastly, artifacts type specification and development information have a low level of impact.

The cross-analysis performed to answer **RQ1** helped us understand which artifacts are more important during the re-engineering. As an exploratory case study, this give us initial evidence about the important of **PAxSPL** generated artifacts. Because this is the

Table 18 – Impact level of artifacts when selecting techniques.

Impact	Artifacts
High	Team Information Report; Requirements Specification;
Medium	Domain Glossary; Domain Constraints List;
Low	Artifacts Type Specification; Development Information Report;

Source: Author.

first time that our proposal is performed in a real development environment, this kind of feedback gives evidence to establish three preliminary conclusions:

1. **No artifact is useless:** all artifacts indicate to contribute somehow to the re-engineering process. Although there are artifacts that are used only in specific phases, they are still useful during that phase.
2. **Collect team information before starting the re-engineering is relevant:** as showed in Table 17 and Table 18, the team information report have high relevance when conducting the re-engineering, specially when selecting retrieval techniques to be used. Despite its importance, this kind of artifact was not present in any of the studies found in the literature (ASSUNÇÃO et al., 2017).
3. **PAxSPL guidelines are very helpful for both the re-engineering, and the retrieval techniques selection:** the guidelines of our proposal were created to give additional support when performing the re-engineering. The collected evidence indicates that they are fulfilling their goal.

In addition, planning the improvement of PAxSPL, we now have evidence that indicates which artifacts we should give more attention. Now, we can plan to focus the activities where these artifacts are generated by, improve their templates, create supporting tools and also indicate to future users which artifacts they could give special treatment.

5.2.4.2 RQ2. What is the quality of the generated artifacts?

This question was created with two objectives: (i) measure the quality of artifacts to indicate that PAxSPL was performed as expected, thus decreasing the chances to invalidate the results of RQ3 (effort). (ii) compare the quality of artifacts generated during both case study phases, analyzing if the results were satisfactory, in order to compare both the time difference (effort), and reusable elements (RQ4).

Considering these two objectives, we could consider RQ2 a secondary research question, used to give support to other two (RQ3 and RQ4). Hence, the analysis presented in this section may be considered an introduction to both following RQs.

As presented in Table 10, during both phases of the case study the generated artifacts achieve satisfactory results, considering our QM. During **Phase 1**, artifacts obtained a total score of 9.5 out of 10. This 0.5 reduction was consequence of the lack of feature artifacts documentation, which in this case were generated with the application of clustering. During **Phase 2**, however, the score was a perfect 10 out of 10.

We can analyze these results considering, individually, the topics of the next questions:

- **Effort (time spent):** the quality of the generated artifacts is important when considering the effort comparison. The reason is because it give us evidence to indicate that independently of the time spent, the process was performed as expected if the quality results were satisfactory. We can consider the artifacts quality both for the each phase of the case study individually, and for comparing them. To individually compare each phase, we have to consider if, despite the time spent in each activity, the artifacts generated by that activity. The artifacts had to obtained a satisfactory quality score, otherwise, it does not really matters in how long time that activity was performed. To compare both phases, however, we have to analyze the quality score difference between them. For instance, **Phase 2** was performed in less than half the time in comparison with **Phase 1**. This result, however, would not be relevant if the artifact quality of **Phase 2** was not satisfactory.
- **Reusability comparison:** for analyzing the reusability of artifacts, the quality is also important. An artifact that indicates being reusable (participants opinion and real time observation) decreases its importance if it does not receive a good score considering its quality.

After analyzing the results of the quality metrics, we could conclude that the comparisons of effort and reusability could be performed without any interference from artifacts with low quality, because for both phases, we achieve good quality results. The analysis for these two topics are detailed in the following sections.

5.2.4.3 RQ3. What is the effort needed to perform each phase of PAXSPL?

The main objective of this RQ is to identify which PAXSPL activities demand more effort. We wanted to collect evidence to compare the activities and phases of our process in terms of effort. This comparison will allow us to improve some aspects of PAXSPL. As we described regarding the artifacts relevance in RQ1, by identifying the activities that demand more effort we can give special attention to them, or even split them in more

activities. The results regarding effort showed in this section are not related to the real effort needed to perform an activity. This measure, however, can be used to analyze the effort difference among the activities.

To calculate the effort, we performed a cross-analysis in the information collected in real time (see Table 9), and the information gathered with the survey answers (see Table 14). We considered the average time spent for each activity during both phases of the case study. We also considered only the average complexity level collected in the survey answers. To merge both results, we multiplied the complexity level by the time spent and called the resulting number as effort score. All this information is summarized in Table 19, showing that the **Execute Assembled Process** activity obtained the highest effort score (12.25) points. **Collect Team Information**, however, obtained the lower score, only 0.25 effort points.

Table 19 – Effort score of each PAXSPL activity.

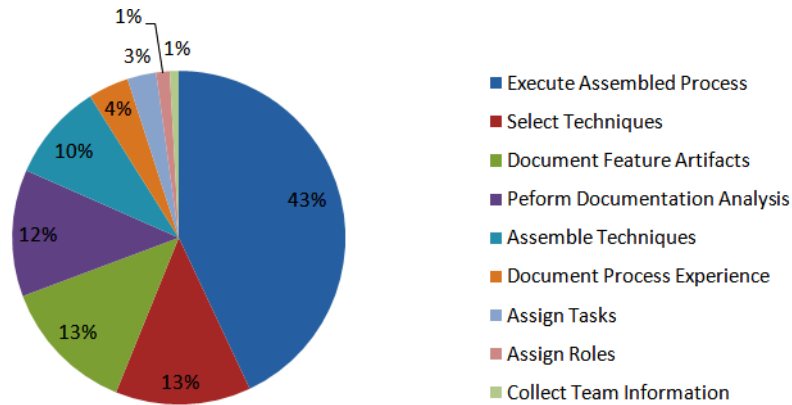
Activity	Time Spent (Hours)	Survey Complexity Score	Effort Score
Collect Team Information	0.25	1	0.25
Assign Roles	0.25	1.5	0.375
Perform Doc. Analysis	1	3.5	3.5
Select Techniques	0.75	5	3.75
Assemble Techniques	0.6	4.5	2.7
Assign Tasks	0.4	2	0.8
Execute Assembled Process	3.5	3.5	12.25
Document Feature Artifacts	1.5	2.5	3.75
Document Process Exp.	0.75	1.5	1.125
Total	9		28.5

Source: Author.

The effort percentage of each activity in relation of the total effort is showed in Figure 20. **Execute Assembled Process** consists of 43% of the total effort. This first number is really expressive because this activity was also pointed as one of the most relevant in RQ1. **Select Techniques** and **Document Feature Artifacts** represent 13% each, while **Perform Documentation Analysis** is 12%, and **Assemble Techniques** consists of 10% of the total effort. These four activities have similar numbers, however, they represent less than a half of **Execute Assembled Process**. By considering the similar results (between 10% and 13%) of the latter activities we can look into splitting **Execute Assembled Process** into more activities that would obtained similar scores.

For the last activities, **Document Process Experience** (4%), **Assign Tasks** (3%), **Assign Roles** (1%) and **Collect Team Information** (1%), we have the opposite situation. These four activities present less than a half effort score than the activities we consider more balanced (between 10% and 13%).

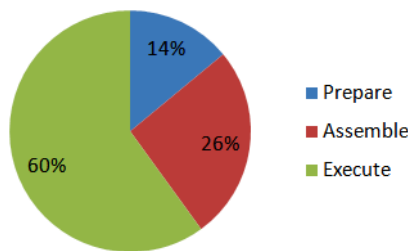
Figure 20 – Effort percentage of each PAXSPL activity.



Source: Author.

By analyzing the total effort of each PAXSPL phase, as illustrated in Figure 21, we see that **Execute** represents 60%, **Assemble** represents 26% and **Prepare** only 14% of the total. This major difference between **Execute** and the others was already expected when we first created PAXSPL. Although we planned the three phases to have similar relevance (as showed in RQ1 answer) the **Execute** is when the feature retrieval is performed so is understandable why its complexity and manly its spent time is so higher than the other two phases.

Figure 21 – Effort percentage of each PAXSPL phase.



Source: Author.

As result of the analysis of this RQ we can formulate two conclusions:

1. **Execute Assembled Process demands more than twice the effort of other activities:** this conclusion give us evidence to plan to perform modification in the **Execute** phase. Although **Execute Assembled Process** effort difference should still be higher, we can look into splitting this activity or reallocating some of its sub-activities to other, aiming to reduce its total effort. We can also plan to develop supporting tools for this activity before the others.
2. **Some activities require almost insignificant effort to be performed:** even if we merge the effort of **Document Process Experience**, **Assign Tasks**, **Assign Roles**

and **Collect Team Information** the result would be lower than other activities. This may indicate that these activities should be reformulated, maybe combined with others or we could even consider to remove some of them from the process and reallocate its generated artifacts to other activities.

In addition, we can include some of this effort information within **PxSPL** guidelines to give future users some basis about how much percentage of the total time they would spent for each activity.

5.2.4.4 RQ4. What is the reuse capability of PxSPL?

To measure the reuse capability (reusability) of **PxSPL** we gave special focus on its generated artifacts. The main objective here is to understand how the re-application of **PxSPL** can be benefited by the reuse of generated artifacts.

We conducted a cross-analysis of information collected in real time (see Table 11) and the survey answers regarding reusability (see Table 15). To calculate the reusability of the artifacts, we considered their condition of use during **Phase 2** which could be: used without modification, updated or not used. If an artifact was used without modification, it would receive 3 reusability points. If an artifact was updated it would receive 1 reusability point. Artifacts not used would receive zero reusability points. These points were added to the reusability score (1 to 5) given by participants in the survey answers. We also gave 2 extra points for artifacts cited in the survey results as more important during **PxSPL** re-application.

Table 20 shows the results of the reusability score calculation. Team information report and retrieval techniques report obtained the maximum possible score, ten points each. Artifacts type specification and feature model of retrieval techniques configured also obtained good scores, eight points each. Requirements specification, with three points, and feature report and domain constraints list, with three and a half points each obtained the lower results.

We created four categories to group the artifacts based on their reusability score:

1. **Not reusable:** artifacts not reused during the **Phase 2** of the case study, despite its reusability score.
2. **Low level of reuse:** artifacts used during **Phase 2** and that scored 4 reusability points or less.
3. **Mid level of reuse:** artifacts used during **Phase 2** and that scored between more than 4 and less than 8 reusability points.
4. **High level of reuse:** artifacts used during **Phase 2** and that scored at least 8 reusability points.

Table 20 – Reusability score of PAXSPL artifacts.

Artifact	Survey Score	Phase 2 Condition Score	Extra Points*	Total Score
Team Information Report	5	3	✓	10
Artifacts Type Specification	5	3		8
Development Information Report	5	0		5
Domain Glossary	3	1		4
Domain Constraints List	2.5	1		3.5
Requirements Specification	2	1		3
Retrieval Techniques Report	5	3	✓	10
FM of Retrieval Tec. Configured	5	3		8
Assembled Process	4.5	1	✓	7.5
Process Documentation	3.5	1		4.5
Feature Report	2.5	1		3.5
Feature Model	4	1		5

* 2 extra points were given to artifacts listed as most important during PAXSPL re-application.

Source: Author.

The results related with this classification are showed in Table 21. We plan to include this information within PAXSPL guidelines to help future users of our process to identify which artifacts they could focus when re-applying PAXSPL.

Table 21 – Reusability level of PAXSPL artifacts.

Reusability Level	Artifacts
High	Team Information Report; Artifacts Type Specification; Retrieval Techniques Report; FM of Retrieval Tec. Configured;
Medium	Assembled Process; Process Documentation; Feature Model;
Low	Domain Glossary; Domain Constraints List; Requirements Specification; Feature Report;
Not Reusable	Development Information Report;

Source: Author.

Based on the results of the analysis performed to answer this RQ, we formulated two conclusions:

1. **The reusability has high impact on the selection of techniques:** among the artifacts that were categorized as having a high level of reuse, we have the retrieval techniques report and feature model of retrieval techniques configured. These two artifacts are both generated during **Select Techniques** activity. As this activity was also found to be the second needing more effort to be performed (see Figure 20),

the reusability of its artifacts was the main reason for the time saved (see Table 9) during **Phase 2** of the case study.

2. **PAXSPL reusability indicates its capability to maybe be applied using the SPL reactive approach:** if we analyze the difference between both phases of the case study, the main difference is the inclusion of new features. This scenario is similar to when the SPL reactive approach is applied in an organization (*e.g.* new requirements arise and new functionalities are implemented). Thus, as showed in (see Table 9), the only activities that demand some time during **Phase 2** of the case study were those when new requirements or features had to be analyzed. Although, our main goal was to create a process to give support to the SPL extractive approach, the reusability results give us initial evidence about PAXSPL benefits when applied in a SPL reactive scenario. This allow us to further investigate the real benefits when using PAXSPL in a reactive approach scenario.

5.2.5 Threats to Validity

In this sub-section we discuss the main threats to the validity of this case study and present how we mitigated these threats based on Runeson e Höst (2009) and Wohlin et al. (2012).

Construct Validity: the first main concern was the misinterpretation of the survey questions by the participants. Aiming to mitigate this problem, we formulated different questions asking the same kind of information and included them in both Survey 2 and 3. We also analyzed the results from all participants and compared if their answers indicated the same level of understandability.

Internal Validity: a possible threat were possible errors when documented information in real time. Trying to mitigate the problem generated by the results of these errors, we analyzed the information collected in real time considering the quality of the artifacts and the survey answers also. With this kind of strategy (triangulation), we could mitigate any problem related with any kind of erroneous or misinterpreted information. Another possible threat was how poor quality of the generated artifacts would impact the results of ours RQ. We mitigated this problem by creating ten QM (see Section 5.2.1.5), and applying in both phases of the case study. The satisfactory results gave us confidence to perform the analysis to answer the RQ.

External Validity: considering the relevance of findings for other cases. We performed, through cross-analysis, different methods to calculate relevance level, reusability and quality of the generated artifacts. We also applied cross-analysis to identify the effort required to perform the activities of PAXSPL. These methods may be used in other cases (*e.g.*, cases evaluating similar proposals).

Reliability: a possible threat are possible problems related with the data depen-

dency related with the researchers conducting the case study. Trying to mitigate this problem, we created the QMs, creating a reliable mechanism to evaluate the generated artifacts. Data triangulation was also used to mitigate possible research bias.

5.3 Chapter Lessons

In this chapter we showed an early validation of our proposal impact, relevance, contribution and flexibility. We could obtain enough confidence to continue our research from the answers. We also identify some points that need improvement, such as flexibility of techniques. We showed and explained the main changes applied in the first version of our process that resulted in the version being presented in this work. We have also conducted and reported a case study to further evaluate our process. The evidence collected will help us improve some aspects of PAXSPL such as its generated artifacts in terms of relevance and reusability. We also identified opportunities for improvement in terms of the effort needed to perform some activities. The analysis results gives additional information to be included in the guidelines of our proposal. For instance, we can indicate to future users which artifacts they should give more attention when deciding which techniques should be used. By summarizing the case study results we can conclude that PAXSPL obtained good results in terms of re-engineering. In addition, it gives evidence about its capability to be applied in a SPL reactive approach.

6 FINAL CONSIDERATIONS

SPL are a systematic way to reuse software products 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 potential to become **SPL**. **SPL** re-engineering processes may have a huge contribution for this field, with techniques, technologies and guidelines to help users to perform feature retrieval.

Through the analysis of a set of **SPL** re-engineering processes mapped by Assunção et al. (2017), we categorized and grouped the most used techniques for feature retrieval. We also created a set of guidelines based on the information collected from those processes. With the objective to create a flexible and customizable process for **SPL** re-engineering we proposed **PxSPL**.

PxSPL is a process that gives support to prepare, assemble and execute feature retrieval in a set of system variants for **SPL** re-engineering. **PxSPL** aims to give enough flexibility for those that may execute it to reduce cost and effort when performing the **SPL** re-engineering process. This flexibility is made possible through team information collection, product documentation analysis, different options of feature retrieval techniques, and process dynamic assembly. We also created a set of guidelines help those that may execute it to decide which techniques should be selected and how to assemble them into our generic process.

To initially evaluate our proposal we applied a survey in experts of the **SPL** re-engineering field. We obtained an early feedback from these experts giving evidence about **PxSPL** contributions and suggesting improvements. Most of the participants agreed that our process provides flexibility, and with the help of our guidelines it may be a relevant contribution to the **SPL** re-engineering field.

In addition, we conducted and reported a case study in a real development environment. The organization where the case study was performed is a startup, called Porthal. Porthal works developing a web based enterprise resource planning system called ICode. ICode its distributed to Porthal's clients in at least ten different configurations (different products) in a no systematic way. These characteristics shows ICode's potential to become **SPL**. We performed **PxSPL** into ten different products generated from ICode, and collect information about: relevance of each **PxSPL** artifact, quality of **PxSPL** generated artifacts, effort spent to perform each **PxSPL** activity, and reuse capability of **PxSPL**.

We divided the case study in two phases, focusing **Phase 2** in the reusability of our process. The analysis of the information collected during the case study gave us evidence to achieve some important conclusions:

1. In regard to the artifacts, they are all useful somehow. Some artifacts indicated to have a high relevance in terms of re-engineering, such as team information report, requirements specification and **PxSPL** guidelines. The results also indicated that

the guidelines our very helpful during the retrieval techniques selection.

2. Considering the quality of the generated artifacts it was satisfactory for both phases of the case study. This information was important because if we obtained bad results in regard with the quality of artifacts, some of our **RQ** would lose relevance.
3. In terms of effort to perform each **PAXSPL** activity, **Execute Assembled Process** received more than twice the effort of the second most effort needed activity. With this result we can plan to split this activity and even to create supporting tools to automatize it. Some activities also indicate to require almost insignificant effort to be performed. In this case, we have to analyze if such activities should be merged to another or even consider to remove them from **PAXSPL**.
4. Considering the reusability, the results indicated that **PAXSPL**, when re-applied in a different but similar scenario, saves time by reusing some of its artifacts. This kind of result give us opportunity to investigate the possible benefits of **PAXSPL** when applied following a reactive **SPL** approach.

Some of the results of the information collected during the case study were already included into **PAXSPL** guidelines.

A preliminary result (the work containing the first version of our process (Section 5.1.4)) was published in the Escola Regional de Engenharia de Software (ERES 2017)¹. We have also submitted part of the work presented here as a paper for the 33rd ACM/SIGAPP Symposium On Applied Computing (SAC 2018)² and for the XXXII Simpósio Brasileiro de Engenharia de Software (SBES)³.

For future work, we plan to continue improving **PAXSPL**. This improvement will be performed by changing, creating and merging some **PAXSPL** activities or sub-processes. In addition we will revise the use and template of its generated artifacts in order to make them more relevant. We also plan to develop supporting tools to automatize and reduce the effort of some activities of our process.

¹ <<https://eventos.unipampa.edu.br/eres/chamada-de-trabalhos/forum-de-graduacao/>>

² <<https://www.sigapp.org/sac/sac2018/>>

³ <http://cbsoft2018.icmc.usp.br/sbes_pt.html>

BIBLIOGRAPHY

- 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. Cited 4 times in pages 38, 39, 40, and 41.
- ACHER, M. et al. On extracting feature models from product descriptions. In: **6th International Workshop on Variability Modeling of Software-Intensive Systems**. [S.l.]: ACM, 2012. p. 45–54. Cited 4 times in pages 38, 39, 40, and 41.
- AL-MSIE'DEEN, A. et al. Feature location in a collection of software product variants using formal concept analysis. 2013. Cited 2 times in pages 58 and 59.
- 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. Cited 2 times in pages 58 and 59.
- ALI, N. et al. Moms: Multi-objective miniaturization of software. In: **27th IEEE International Conference on Software Maintenance**. [S.l.]: IEEE, 2011. p. 153–162. Cited in page 57.
- 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. Cited 3 times in pages 56, 59, and 60.
- ARAAR, I. E.; SERIDI, H. Software features extraction from object-oriented source code using an overlapping clustering approach. **Informatika (Slovenia)**, Slovenian Society Informatika/Slovensko drustvo Informatika, v. 40, n. 2, p. 245–255, 2016. Cited in page 38.
- ASSUNÇÃO, W. et al. Reengineering legacy applications into software product lines: a systematic mapping. **Empirical Software Engineering**, Springer, p. 1–45, 2017. Cited 13 times in pages 21, 22, 25, 28, 29, 33, 34, 35, 37, 41, 49, 90, and 99.
- ASSUNÇÃO, W. K. et al. Multi-objective reverse engineering of variability-safe feature models based on code dependencies of system variants. **Empirical Software Engineering**, Springer, p. 1–32, 2016. Cited in page 38.
- ASSUNÇÃO, W. K. G. ModelVars2SPL: an automated approach to reengineer model variants into software product lines. 2017. Cited in page 38.
- BAYER, J. et al. Definition of reference architectures based on existing systems. **IESE-Report**, Citeseer, 2004. Cited in page 57.
- BÉCAN, G. et al. Breathing ontological knowledge into feature model management. 2013. Cited 6 times in pages 30, 38, 39, 40, 41, and 56.
- 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. Cited in page 56.
- CHIKOFSKY, E.; CROSS, J. Reverse engineering and design recovery: A taxonomy. **IEEE software**, IEEE, v. 7, n. 1, p. 13–17, 1990. Cited 2 times in pages 21 and 28.

- CHRISTENSEN, A.; MØLLER, A.; SCHWARTZBACH, M. Precise analysis of string expressions. **Static Analysis**, Springer, p. 1076–1076, 2003. Cited in page 28.
- CLEMENTS, P.; NORTHROP, L. **Software product lines**. [S.l.]: Addison-Wesley, 2002. Cited 2 times in pages 25 and 30.
- 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. Cited in page 31.
- DAMAŠEVIČIUS, R. 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. Cited in page 56.
- DAVIS, S. From “future perfect”: Mass customizing. **Planning review**, MCB UP Ltd, v. 17, n. 2, p. 16–21, 1989. Cited in page 25.
- DUMAIS, S. Latent semantic analysis. **Annual review of information science and technology**, Wiley Online Library, v. 38, n. 1, p. 188–230, 2004. Cited in page 30.
- EISENBARTH, T.; KOSCHKE, R.; SIMON, D. Derivation of feature component maps by means of concept analysis. In: **Software Maintenance and Reengineering, 2001. Fifth European Conference on**. [S.l.]: IEEE, 2001. p. 176–179. Cited in page 58.
- 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. Cited 2 times in pages 58 and 59.
- EYAL-SALMAN, H.; SERIAI, D.; DONY, C. Identifying traceability links between product variants and their features. In: **1st International workshop on Reverse Variability Engineering**. [S.l.: s.n.], 2013. p. 17–22. Cited 2 times in pages 59 and 60.
- 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. Cited 2 times in pages 56 and 58.
- 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. Cited in page 31.
- FRAKES, W.; BAEZA-YATES, R. Information retrieval: data structures and algorithms. Prentice Hall PTR, 1992. Cited in page 29.
- FUGGETTA, A.; NITTO, E. D. Software process. In: ACM. **Proceedings of the on Future of Software Engineering**. [S.l.], 2014. p. 1–12. Cited in page 32.
- JAIN, A. K.; DUBES, R. C. **Algorithms for clustering data**. [S.l.]: Prentice-Hall, Inc., 1988. Cited in page 29.
- KANG, K. et al. **Feature-oriented domain analysis (FODA) feasibility study**. [S.l.], 1990. Cited 3 times in 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. Cited 2 times in pages 22 and 25.
- KELLY, M. et al. Recovering a balanced overview of topics in a software domain. In: **Source Code Analysis and Manipulation (SCAM), 2011 11th IEEE International Working Conference on**. [S.l.]: IEEE, 2011. p. 135–144. Cited in page 56.
- KITCHENHAM, B. et al. Systematic literature reviews in software engineering—a systematic literature review. **Information and software technology**, Elsevier, v. 51, n. 1, p. 7–15, 2009. Cited in page 33.
- 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. Cited 2 times in pages 29 and 57.
- KRUCHTEN, P. **The rational unified process: an introduction**. [S.l.]: Addison-Wesley Professional, 2004. Cited in 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. Cited 3 times in pages 21, 22, and 27.
- KULESZA, U. et al. Mapping features to aspects: A model-based generative approach. In: **Early Aspects Workshop**. [S.l.]: Springer, 2007. p. 155–174. Cited in page 38.
- KUMAKI, K. et al. Supporting commonality and variability analysis of requirements and structural models. In: **16th International Software Product Line Conference—Volume 2**. [S.l.]: ACM, 2012. p. 115–118. Cited 2 times in pages 38 and 60.
- 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. Cited in page 21.
- LIKERT, R. A technique for the measurement of attitudes. **Archives of psychology**, 1932. Cited in page 65.
- 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 3 times in pages 21, 25, and 26.
- LINSBAUER, L. et al. Recovering feature-to-code mappings in mixed-variability software systems. In: **Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on**. [S.l.]: IEEE, 2014. p. 426–430. Cited in page 57.
- MAAZOUN, J.; BOUASSIDA, N.; BEN-ABDALLAH, H. A bottom up spl design method. In: **Model-Driven Engineering and Software Development (MODELSWARD), 2014 2nd International Conference on**. [S.l.]: IEEE, 2014. p. 309–316. Cited 2 times in pages 58 and 59.

- MAÂZOUN, J.; BOUASSIDA, N.; BEN-ABDALLAH, H. Feature model recovery from product variants based on a cloning technique. In: **Software Engineering and Knowledge Engineering**. [S.l.: s.n.], 2014. p. 431–436. Cited 2 times in pages 58 and 59.
- MARTIN, R. C. **Agile software development: principles, patterns, and practices**. [S.l.]: Prentice Hall, 2002. Cited in page 32.
- 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. Cited 6 times in pages 22, 25, 38, 39, 40, and 41.
- MEFTEH, M.; BOUASSIDA, N.; BEN-ABDALLAH, H. Feature model extraction from documented uml use case diagrams. **ADA USER**, v. 35, n. 2, p. 107, 2014. Cited in page 58.
- MERSCHEN, D. et al. Experiences of applying model-based analysis to support the development of automotive software product lines. In: **5th Workshop on Variability Modeling of Software-Intensive Systems**. [S.l.]: ACM, 2011. p. 141–150. Cited in page 38.
- 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. Cited 2 times in pages 30 and 61.
- NÖBAUER, M.; SEYFF, N.; GROHER, I. Inferring variability from customized standard software products. In: **International Software Product Line Conference-Volume 1**. [S.l.]: ACM, 2014. p. 284–293. Cited in page 57.
- NÖBAUER, M.; SEYFF, N.; GROHER, I. Similarity analysis within product line scoping: An evaluation of a semi-automatic approach. In: **International Conference on Advanced Information Systems Engineering**. [S.l.]: Springer, 2014. p. 165–179. Cited in page 56.
- 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. Cited 2 times in pages 22 and 25.
- PETERSEN, K. et al. Systematic mapping studies in software engineering. In: **EASE**. [S.l.: s.n.], 2008. v. 8, p. 68–77. Cited in page 33.
- 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 in pages 21, 25, and 26.
- POLZER, A. et al. Managing complexity and variability of a model-based embedded software product line. **Innovations in Systems and Software Engineering**, Springer, v. 8, n. 1, p. 35–49, 2012. Cited in page 38.

- PRESSMAN, R. S. **Software engineering: a practitioner's approach**. [S.l.]: Palgrave Macmillan, 2005. Cited in page 32.
- RUBIN, J.; CHECHIK, M. Combining related products into product lines. In: **Inter. Conf. on Fundamental Approaches to Software Engineering**. [S.l.]: Springer, 2012. p. 285–300. Cited in page 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. Cited 2 times in pages 30 and 61.
- RUBIN, J. et al. Managing forked product variants. In: **16th International Software Product Line Conference-Volume 1**. [S.l.]: ACM, 2012. p. 156–160. Cited in page 56.
- RUNESON, P.; HÖST, M. Guidelines for conducting and reporting case study research in software engineering. **Empirical software engineering**, Springer, v. 14, n. 2, p. 131, 2009. Cited in page 96.
- RUNESON, P. et al. **Case study research in software engineering: Guidelines and examples**. [S.l.]: John Wiley & Sons, 2012. Cited in page 73.
- 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. Cited in page 29.
- 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. Cited in 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. Cited 4 times in pages 38, 39, 40, and 41.
- SCHWABER, K.; BEEDLE, M. **Agile software development with Scrum**. [S.l.]: Prentice Hall Upper Saddle River, 2002. v. 1. Cited in page 32.
- 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. Cited 2 times in pages 22 and 25.
- STUMME, G. Formal concept analysis. In: **Handbook on ontologies**. [S.l.]: Springer, 2009. p. 177–199. Cited in page 29.
- TANG, Y.; LEUNG, H. Top-down feature mining framework for software product line. In: **ICEIS (2)**. [S.l.: s.n.], 2015. p. 71–81. Cited in page 38.
- WESTON, N.; CHITCHYAN, R.; RASHID, A. A framework for constructing semantically composable feature models from natural language requirements. In: **13th International Software Product Line Conference**. [S.l.]: Carnegie Mellon University, 2009. p. 211–220. Cited in page 56.
- WOHLIN, C. et al. **Experimentation in Software Engineering**. [S.l.]: Springer, 2012. 236 p. Cited in page 96.

XUE, Y.; XING, Z.; JARZABEK, S. Feature location in a collection of product variants. In: **Reverse Engineering (WCRE), 2012 19th Working Conference on**. [S.l.]: IEEE, 2012. p. 145–154. Cited 2 times in pages [58](#) and [59](#).

YANG, Y.; PENG, X.; ZHAO, W. Domain feature model recovery from multiple applications using data access semantics and formal concept analysis. In: **Reverse Engineering, 2009. WCRE'09. 16th Working Conference on**. [S.l.]: IEEE, 2009. p. 215–224. Cited in page [58](#).

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. Cited 2 times in pages [22](#) and [25](#).

Appendix

APPENDIX A – TEMPLATE FOR TEAM INFORMATION REPORT

Figure 22 – Template for Team Information Report

This is a Template Document and may be modified as needed

Team Information Report

Created by: Team Member 0

Date:

Member:	<i>Team Member 1</i>
Email:	<i>email@email.com</i>
Roles in company:	<i>Analyst, Developer</i>
Roles during product development:	<i>Analyst, Developer, Tester</i>
Experience working with SPL	<i>Participated in several SPL development processes.</i>
Knowledge about retrieval techniques	<i>Applied Formal Concept Analysis in some projects.</i>
Obs:	<i>Domain Specialist</i>

Member:	<i>Team Member 2</i>
Email:	<i>email@email.com</i>
Roles in company:	<i>Analyst</i>
Roles during product development:	<i>Analyst, Tester</i>
Experience working with SPL	<i>None</i>
Knowledge about retrieval techniques	<i>Applied data-flow analysis algorithms in some projects.</i>
Obs:	<i>Specialist in requirements</i>

Member:	<i>Team Member N</i>
Email:	<i>...</i>
Roles in company:	<i>...</i>
Roles during product development:	<i>...</i>
Experience working with SPL	<i>...</i>
Knowledge about retrieval techniques	<i>...</i>
Obs:	<i>...</i>

APPENDIX B – TEMPLATE FOR ARTIFACTS TYPE SPECIFICATION

Figure 23 – Template for Artifacts Type Specification

This is a Template Document and may be modified as needed.

Artifacts Type Specification

Created by: Team Member 0

Date:

Product	<i>Product 1</i>	
Types of artifacts	<i>Requirements list, Class diagrams</i>	
Artifacts Description		
<i>Requirements List</i>	Type	<i>Textual artifact</i>
	Overview:	<i>Description of functional</i>
	Extension	<i>.pdf</i>
<i>Class Diagrams</i>	Type	<i>Graphic artifacts</i>
	Overview:	<i>Diagrams of the software classes showing relationships and attributes</i>
	Extension	<i>.png</i>

Product	<i>Product 2</i>	
Types of artifacts	<i>Use Cases</i>	
Artifacts Description		
<i>Use Cases</i>	Type	<i>Textual artifact</i>
	Overview:	<i>Description of functional and non functional requirements</i>
	Extension	<i>.doc</i>

Product	<i>Product N</i>	
Types of artifacts	<i>...</i>	
Artifacts Description		
<i>...</i>	Type	<i>...</i>
	Overview:	<i>...</i>
	Extension	<i>...</i>

APPENDIX C – TEMPLATE FOR RETRIEVAL TECHNIQUES REPORT

Figure 24 – Template for Retrieval Techniques Report

This is a Template Document and may be modified as needed

Retrieval Techniques Selection

Created by: Team member N

Date:

Candidate Technique:	<i>Technique X</i>
Related Techniques:	<i>Y, Z</i>
Selection Result	<i>Selected (or not selected)</i>
Reasons	<i>....</i>
Obs:	

APPENDIX D – TEMPLATE FOR FEATURE REPORT

Figure 25 – Template for Feature Report

This is a template document and may be modified as needed.

Feature Report

Created by: Team member 1

Date:

Feature Name:	<i>F1</i>
Category:	<i>Mandatory</i>
Parent:	<i>Root</i>
Constraint:	<i>$F1 \Rightarrow F2 \wedge F3$</i>
Traceability:	
Document Name:	<i>Requirements List</i>
Document Type:	<i>Text</i>
Entry Point Description:	<i>Requirements 1 and 5</i>
Finding Method:	<i>FCA application</i>

APPENDIX E – TEMPLATE FOR PROCESS EXECUTION REPORT

Figure 26 – Template for Process Execution Report

This is a Template Document and may be modified as needed

Process Execution Report

Created by: Team Member 0

Date:

Activity:	<i>Collect Team Information</i>
Team Member Responsible:	<i>Member 1</i>
Decisions:	<i>It was decided to...</i>
Difficulties:	<i>None</i>
Approximate Time Spent:	<i>5 hours</i>
Obs:	<i>....</i>

Activity:	<i>Assign Roles</i>
Team Member Responsible:	<i>Member 2</i>
Decisions:	<i>It was decided to...</i>
Difficulties:	<i>None</i>
Approximate Time Spent:	<i>5 hours</i>
Obs:	<i>...</i>

INDEX

EC, 34, 37

FCA, 29, 46, 55–60, 77, 78

FODA, 30

IC, 34, 37

LSI, 30, 55, 59, 60, 77

PAxSPL, 43, 46, 55, 61, 63, 65, 68, 70–78,
83, 85–91, 93–97, 99, 100

QA, 33, 35, 37, 41

QM, 74, 75, 77, 91, 96, 97

RQ, 34, 37, 71, 74–76, 85–87, 89–93, 95,
96, 100

SLR, 33, 34, 37, 38, 41

SMS, 33, 34, 41

SPL, 21–23, 25–28, 30, 32–35, 41, 43, 45,
55, 61, 63, 65–69, 72, 75, 77, 85,
96, 97, 99, 100

SPLA, 27

SPLE, 21–23, 25, 26, 32, 40, 65

VSM, 30, 55, 60