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

Luciano Marchezan[1], **Elder Rodrigues**[2], **Maicon Bernardino**[2]

[1]ISSE - Johannes Kepler University Linz Linz – Austria

lucianomarchp@gmail.com

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

{elderrodrigues,bernardino}@unipampa.edu.br

Abstract. *Extractive Software Product Line (SPL) is a well-known approach that organizations can use to transform their legacy applications into an SPL. In this sense, the SPL reengineering process emerges as a possible strategy for applying the extractive SPL. Available artifacts used to perform the SPL reengineering may change, requiring software engineers to adapt their approaches as a means to satisfying the companies' scenarios. However, there is a lack of an approach supporting this adaptation considering different scenarios. To address these limitations we propose the Prepare, Assemble, and Execute Framework SPL reengineering (PAxSPL). PAxSPL is composed of three different aspects: a process, guidelines, and a supporting tool. For evaluating PAxSPL, we extracted eight different scenarios from the literature. Results evidenced that PAxSPL is customizable to a variety of scenarios with different reengineering artifacts, techniques, and activities.*

Resumo. *A abordagem de Linhas de Produto de Software (SPL) extrativa pode ser usada por organizações que pretendem transformar seus sistemas legados em SPL. Nesse contexto, o processo de reengenharia de SPL é uma possível estratégia para se aplicar esta abordagem. Os artefatos disponíveis usados para se realizar a reengenharia de SPL podem mudar, exigindo que os engenheiros de software adaptem suas abordagens para se adequar aos cenários das empresas. No entanto, identificamos a falta de uma abordagem que de suporte a esta adaptação considerando diferentes cenários. Para atender essa limitação, propomos o Prepare, Assemble and Execute framework para reengenharia de SPL (PAxSPL). PAxSPL é composto por três partes diferentes: um processo, diretrizes e uma ferramenta de suporte. Para avaliar o PAxSPL, extraímos oito cenários diferentes da literatura. Os resultados evidenciaram que o PAxSPL é customizável para uma variedade de cenários com diferentes artefatos, técnicas e atividades de reengenharia.*

1. Introduction

Software development companies must ensure the quality of software products to meet user's needs and remain competitive. Yet, these companies also have to find ways to reduce development time and costs [Krüger et al. 2020]. This trade-off leads companies to

look for solutions such as software Product Line Engineering (SPLE). SPL is an industry-proven software development approach that relies on software reuse to enable companies to more effectively and efficiently develop their product portfolios [Pohl et al. 2005]. This software reuse can be based, for example, on existing artifacts, assets, or expertise of developers [Capilla et al. 2019]. The most common way to adopt SPLE is by employing an extractive strategy [Krüger et al. 2020]. Companies usually have many system variants developed using opportunistic reuse, which is the basis for the SPLE activities [Assunção et al. 2017]. Extractive adoption of Software Product Lines (SPL) is conducted by a reengineering process that encompasses the identification and extraction of common and variable features [Laguna and Crespo 2013]. This process is divided into three phases, namely detection, analysis, and transformation [Assunção et al. 2017]. A proper planning[1] of all these steps is paramount for the success of the SPL adoption.

Despite the great number of studies on the topic of SPL reengineering [Assunção et al. 2017, Laguna and Crespo 2013], we observed limitations that motivated us to propose PAxSPL. Firstly, *the studies in the literature mostly focus on technical aspects of the variants*, namely identification of commonalities and variabilities among variants implementation [Eyal-Salman et al. 2013, Paškevičius et al. 2012], without taking into account organizational aspects, *e.g.*, team experience. Considering only technical aspects hampers the application of the SPL reengineering process in real-world scenarios. Secondly, among existing pieces of work, we could see that *the vast majority of approaches are inflexible and strictly designed for a specific scenario*, hampering their use in other companies and systems. Thirdly, *existing automated support is mainly based on academic prototypes developed for validating an approach, not mature enough to be used in practice* [Laguna and Crespo 2013]. These tool prototypes, although important to be used as a proof-of-concept, are not designed with a focus on the end-user, limiting their application in real scenarios.

Based on the aforementioned limitation of existing approaches, in this work, we present the Prepare Assemble and Execute Framework for SPL Reengineering (PAxSPL). PAxSPL framework is an evolution of the PAxSPL process [Marchezan et al. 2019], adding scoping specific information collected through the execution of a Systematic Literature Review (SLR), as well as the development of a supporting tool. PAxSPL was evaluated with eight distinct scenarios extracted from the ESPLA catalog [Martinez et al. 2017]. We instantiated these scenarios using technical and organizational aspects to better understand the benefits and drawbacks of our framework. The results of the evaluation indicate that our framework supports the planning of a variety of SPL reengineering scenarios, considering different artifacts (more than 20 types), techniques (at least five), and activities. The results also evidence that our framework organizes and generates a repository of documents containing both technical and organizational information that might be important for the SPL reengineering analysis and evolution. We also identified eight challenges, that were used to discuss the lessons learned when conducting the evaluation.

The remainder of this work is structured as follows: Section 2 presents the back-

[1]In this work, SPL reengineering planning refers to the activities for initiating, instantiating, documenting, and analyzing the activities, techniques, methods, and tools to be used along all the reengineering process.

ground with the main concepts from our research. Section 3 presents the PAxSPL framework and its supporting tool. Section 4 presents the evaluation performed. Section 5 concludes this work.

2. Background

In this section, we describe the main concepts related to our work.

SPL is defined by [Clements and Northrop 2002] as a set of software-intensive systems that share a common and managed set of features satisfying the specific needs of a particular market and that are developed from a common set of core assets in a prescribed way. As defined by [Pohl et al. 2005], "SPLE is the paradigm responsible for the development and study of SPLs. It uses platforms and mass customization concepts to enable variability management". There are three possible approaches for an organization moving from traditional software to SPL: proactive, reactive, and extractive. The extractive approach is the most indicated when the organization already has a set of software variants because it allows to identify and extract the commonalities and the variabilities among them. This process of extraction is called SPL Reengineering [Assunção et al. 2017].

In the SPL context, reengineering is used to transform a system, system family, or system variants into an SPL. According to [Assunção et al. 2017], the SPL reengineering process is composed of three main phases: detection, analysis, and transformation. During the first phase, detection, variability, and commonalities of the products are identified and extracted from the system variants through the use of feature retrieval techniques. These characteristics are represented in the form of features. Techniques and methods used during this phase aim to extract data from artifacts, such as class diagrams and source code. The second phase is analysis, where the discovered features are organized as a feature model. The feature model is the most used mechanism to represent SPL variability [Clements and Northrop 2002]. In the SPL context, the feature model is represented in a tree structure, where its root is usually the SPL being modeled. The last phase, transformation, is when artifacts linked with these features are managed and modified to create the SPL.

In the context of SPL reengineering, planning also has to consider costs and business goals. A plan to apply SPL-based techniques considering business alignment and goals is called scoping [John 2010]. SPL scoping aids companies to define the boundaries of their products. Literature abounds with proposals for SPL-based production plans as surveyed in [John and Eisenbarth 2009]. These works are devoted to characterizing particular artifacts and tasks for specific domains. Proposals such as PuLSE are well-known in the literature and despite being developed for being customizable, the approach does not provide guidance for adaptability for a reengineering context.

3. The PAxSPL framework

In this section, we describe the evolution from past work, as well as how PAxSPL[2] customization is possible. We also explain how we handle SPL scoping in the SPL reengineering context.

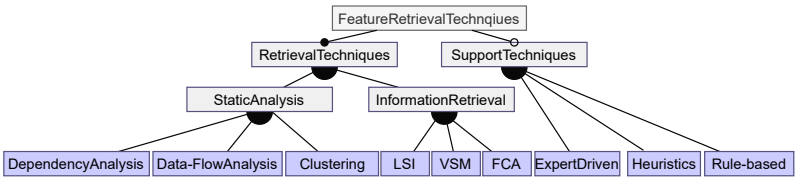
[2]Further details are found in our framework repository at `https://github.com/HestiaProject/PAxSPL/`

3.1. Evolution from Past Work

To evolve the PAxSPL process [Marchezan et al. 2019], we conducted a SLR[3] to investigate SPL scoping approaches. We identified, analyzed, and extracted important information from SPL scoping works found in the literature. The search was carried out in six digital libraries, as well as by performing snowballing. With the results of the search, and selection of studies by applying inclusion/exclusion and quality criteria, we analyzed a total of 44 studies, identifying 32 different proposals. This analysis allowed us to identify the major approaches in the field, establishing a feature model of SPL scoping concepts (Figure 3) and a generic scoping process (Figure 4). This information includes similarities and differences among these approaches as well as scoping concepts and research opportunities. All this information was included in PAxSPL guidelines. Furthermore, we developed a supporting tool to guide users when performing the process. The requirements of the supporting tool were defined based on PAxSPL activities, limitations found in the literature [Assunção et al. 2017] and the analysis of related tools [Pereira et al. 2015]. PAxSPL framework, including its process, guidelines, and tool are discussed next.

3.2. Customization for Different Scenarios

The Prepare, Assemble, and Execute framework for SPL Reengineering (PAxSPL) was designed for giving its users enough guidance when conducting feature retrieval of legacy systems. As organizational contexts change, the approach must cover these changes. Thus, we defined guidelines with alternatives techniques and strategies for performing the retrieval. As illustrated in Figure 1, we grouped the techniques based on their strategy. We have the mandatory group of Retrieval Techniques which are composed of two Or-alternatives (at least one must be selected), Static Analysis, *e.g.*, dependency analysis, and Information Retrieval techniques, *e.g.*, clustering. The second group is optional, composed of three techniques: Expert Driven Extraction, Rule-Based Techniques, and Heuristics. These techniques should be executed only to improve the result of the retrieval strategies of the first group.



LSI - Latent Semantic Indexing; VSM - Vector Space Model; FCA - Formal Concept Analysis.

Figure 1. A Feature Model of Retrieval Techniques.

Based on the selection of the techniques, the user would assemble them into the generic process for feature retrieval, shown in Figure 2. The process is composed of four basic activities. An example of an assembled process would be using Formal Concept Analysis (FCA) during the *Extract* task, Latent Semantic Indexing (LSI) for the *Categorize* and clustering for *Group*. Then, the feature artifacts would be converted to a feature model.

[3]This SLR is currently under review by an international journal. The complete SLR protocol and results are found in Chapter 4 of our thesis [Marchezan 2020].

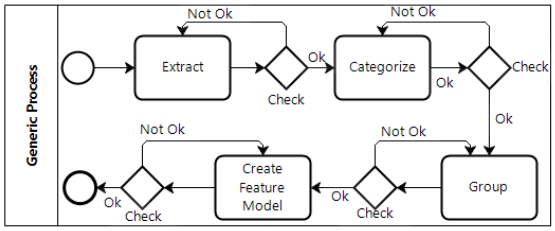


Figure 2. The Generic Process for Feature Retrieval.

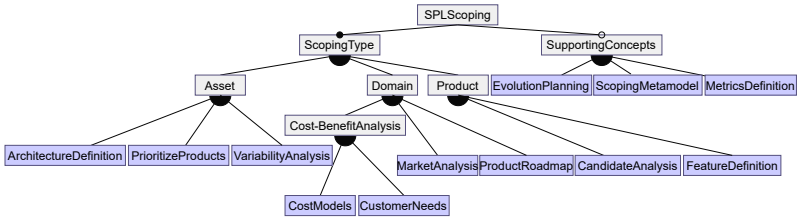


Figure 3. A feature Model of SPL Scoping Concepts

In addition to the feature retrieval, PAxSPL also guides customization considering the SPL scoping. As a result of the aforementioned SLR, we were able to establish a feature model of scoping concepts. Figure 3 presents these concepts which are divided by Scoping type, *e.g.*, architecture definition, and Supporting concepts, *e.g.*, evolution planning.

Similar to the feature retrieval strategies, the scoping concepts must be selected by PAxSPL's users and assembled into a generic SPL Scoping process, presented in Figure 4. *Pre-Scoping* is the first task, where supporting concepts from the Scoping feature model may be used, such as metrics definition. Then, the users would assemble the activities related to which scoping type. The selection of activities is performed based on the user's context. For instance, a market analysis may be performed in the *domain scoping* activity. As presented in the BPMN model, the activities related to the scoping type are not mandatory, however, at least one must be performed. Lastly, the *scoping closure* activity is executed. This activity is generic as several ways to close the scoping process may be performed, such as evolution planning.

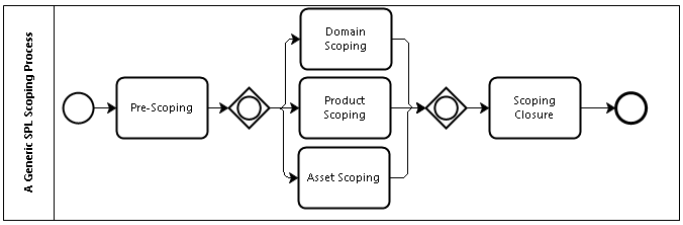


Figure 4. A Generic SPL Scoping Process.

3.3. PAxSPL Process

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

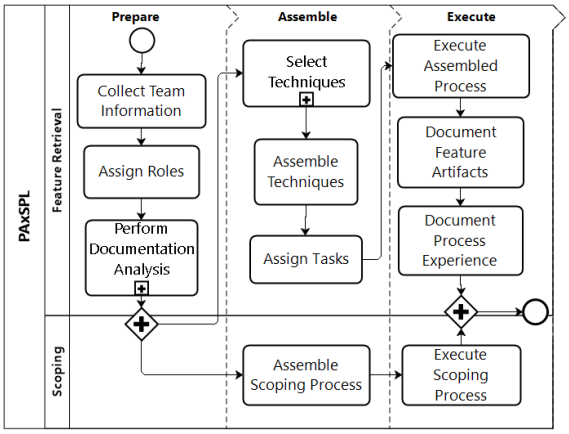


Figure 5. PAxSPL process workflow.

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

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

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

3.4. Guidelines and Supporting tool

By analyzing other SPL reengineering approaches, mapping the used strategies and artifacts, we created a set of guidelines. These guidelines contain support documentation to help to choose techniques, describe each one, give examples, supporting tools, define recommended scenarios and give a prioritization assemble order when assembling a technique into the generic process. Among the many information found in the guidelines section of our process, we included a basic introduction to SPL and its main concepts. We also provided information about variability management and feature model notations and tools. These guidelines are important for selecting both the feature retrieval techniques and scoping concepts. We aimed at giving support to newcomers, however, the guidelines may also aid experienced practitioners.

These guidelines were implemented into our supporting tool[4]. The tool was developed following an iterative development life-cycle. For that, we performed three iterations, on which we defined requirements based on the following: i) automated support to PAxSPL activities [Marchezan et al. 2019]; ii) address limitations evidenced in the literature [Marchezan et al. 2019, Assunção et al. 2017, Laguna and Crespo 2013], and iii) end-user requirements of similar tools [Pereira et al. 2015]. More details related to our tool are available in Chapter 5 in our thesis [Marchezan 2020].

4. Evaluation

We performed a study to evaluate PAxSPL. In this section, we describe the goal, research questions (RQs), the dataset used, and the execution of this study.

4.1. Goal and Research Questions

The main goal of our study is to *evaluate how PAxSPL supports and automates the SPL reengineering planning for different scenarios*. Thus, we aim to apply PAxSPL to customize different reengineering processes in scenarios observed in related works. The RQs that guided the evaluation are:

RQ1. Is PAxSPL customizable to different scenarios?

RQ2. How does PAxSPL suit different scenarios?

RQ3. What challenges are observed by customizing PAxSPL?

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

[4]Our tool is open source and its documentation, including user stories, is available at `https://tinyurl.com/paxspltool`

4.2. Dataset and Execution

The dataset used as input for this evaluation is composed of studies that report on the execution of an SPL reengineering process. We selected studies mapped in the ESPLA catalog [Martinez et al. 2017], which is a collaborative catalog of case studies on SPL reengineering. We randomly selected the studies from the catalog, applying three criteria in each of them, for deciding to include it or not. The study should be approved in all the following three criteria to be selected for this evaluation: i) *The study applies at least one retrieval technique supported by our framework*: we only considered studies that applied at least one retrieval technique among those covered by PAxSPL.[5] For example, in a study that applies three techniques, if one of them is covered by PAxSPL, this criteria is satisfied, even that the other two techniques are not supported. ii) *The study presents a different scenario from other studies previously selected*: we want to guarantee that all selected studies have different scenarios from each other. We considered different scenarios when the study used at least one different retrieval technique, used at least one different input artifact, or had a different workflow when applying the feature retrieval techniques. iii) *The study evaluation protocol, dataset, and results are available online*: Thus, we could extract their activities, artifacts, and techniques to conduct our evaluation.

After applying these criteria to 23 studies randomly selected from the ESPLA catalog, we composed a final set of eight studies, to be used as input for PAxSPL evaluation.[6] Table 1 presents the data extracted from these studies. There is a variety of different artifacts from the studies, ranging from domain to development artifacts. Also, different techniques were used in these scenarios. There is also a different combination of techniques, which defines a different scenario. We executed PAxSPL in each scenario, *i.e.*, the artifacts, retrieval techniques, and activities, from the selected studies. Hence, once we started PAxSPL execution for one scenario, we intend to observe if we can fully represent that scenario. We focus on evaluating how PAxSPL automates the planning of the SPL reengineering process, comparing to the original process observed in the original study. For each study selected, we performed the following steps: i) identify and register inputs and output artifacts; ii) identify and register feature retrieval techniques used; iii) identify the feature retrieval activities and their workflow; and iv) execute PAxSPL using the artifacts, techniques, and activities identified.

After executing these steps, we collected information to answer the RQs by analyzing: 1. The number of artifacts from the original study that were used by PAxSPL (**RQ1 and RQ2**); 2. The retrieval techniques that were instantiated into PAxSPL generic process (**RQ1 and RQ2**); 3. The activities that were instantiated into PAxSPL generic process (**RQ1 and RQ2**); 4. The number of scenarios for which PAxSPL was able to adapt to (**RQ2**); 5. The challenges and limitations found when adapting the scenarios using PAxSPL (**RQ3**).

4.3. Results and Analysis

In the following, we present the results and two artifacts generated by our tool during this execution. Due to space limitations, we only present examples of the artifacts for the first scenario. All artifacts are available in Chapter 6 of our thesis [Marchezan 2020].

[5]PAxSPL supports the same retrieval techniques presented in [Marchezan et al. 2019].

[6]Selected studies/scenarios, and evaluation artifacts are available in `https://doi.org/10.5281/zenodo.4024524`.

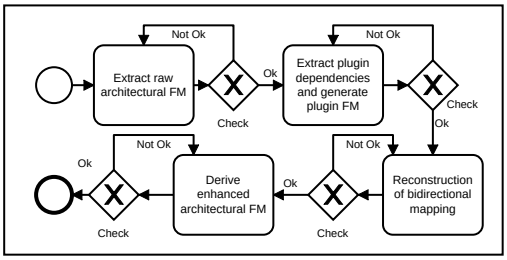
Table 1. Data Extracted from the Original Studies

Reference	Artifacts	Techniques	Activities
[Eyal-Salman et al. 2013]	object-oriented source code; feature descriptions	LSI; FCA; clustering;	i) Use LSI to divide features and classes into common and variable partitions; ii) fragment variable partitions into minimal disjoint sets using FCA; iii) derive code-topics from common class partition; iv) perform traceability links between features and their code-topics; v) determine which classes implement each feature.
[Acher et al. 2013]	150% architecture of the system; specification of the system plugins; system plugins dependencies; architectural FM; plugin FM; constraints mapping; enhanced architectural FM;	dependency analysis; structural similarity; clustering;	i) extraction of a raw architectural FM; ii) extraction of plugin dependencies to derive inter-feature constraints from inter-plugin constraints (plugin FM); iii) automatically reconstruction of the bidirectional mapping between the architect FM and plugin FM; iv) explore the mapping to derive enhanced architectural FM.
[Al-Msie'Deen et al. 2012]	object-oriented source code; object-oriented building elements; commonalities and variations; blocks of variations; atomic blocks of variation;	LSI; FCA;	i) analyze OO source code to extract OO building elements; ii) commonalities and variations are extracted using FCA (blocks of variations); iii) blocks of variations are divided into atomic blocks and features are identified based on textual similarity using FCA and LSI.
[Shatnawi et al. 2014]	object-oriented source code; component architecture; sets of component variants; concept lattice; architecture variability;	dependency analysis; FCA; ROMANTIC approach;	i) extract component-based architecture; ii) identity component variants; iii) use FCA to analyze the commonality and variability; iv) identify architecture variability.
[Alves et al. 2008]	requirement documents; requirements clusters; configurations; feature model;	LSI; Vector Space Model; clustering;	i) perform a requirements similarity determination; ii) abstract requirements clusters into a configuration; iii) merge configurations for all requirements.
[Chen et al. 2005]	individual requirements; requirements relationship graph; application feature trees; domain feature tree;	clustering;	i) requirements elicitation; ii) requirements relationship graph construction; iii) requirements clustering and hierarchical structure construction; iv) merging and variability modeling.
[Paškevičius et al. 2012]	java source code; java .class files; dependency graph; feature distance matrix; cluster dendrogram; feature model; Feature model defined in FDL/Prolog;	dependency analysis; clustering;	i) compile Java source code using a standard Java compiler ; ii) extract feature dependencies from Java class files; iii) construct a feature distance matrix; iv) cluster features based on their dependency in a feature tree; v) convert a feature tree into a FM; vi) generate description of FM in FDL/Prolog.
[Breivold et al. 2008]	architecture description; design documents; source code; user documentation; requirements specification; architecture requirements; common core assets; variable assets; SPL architecture;	dependency analysis;	i) identify requirements on the software architecture; ii) identify commonalities and variabilities; iii) restructure architecture; iv) incorporate commonality and variability; v) evaluate software architecture quality attributes.

4.4. Scenario's Results

The first scenario was extracted from [Eyal-Salman et al. 2013]. Figure 6(a) presents the BPMN[7] representation of the instantiated retrieval process. The five activities and their workflow were based on the original study, however, minor modifications were made due to some challenges faced (see Section 4.5). As the first scenario of the evaluation, this was unique in all aspects as no artifacts, techniques, and activities were previously executed. PAxSPL gave support to the documentation of the different types of artifacts

[7] All BPMN diagrams and reports were generated by our tool.



(a) BPMN Process Instantiated for [Eyal-Salman et al. 2013]

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

(b) Report excerpt for [Eyal-Salman et al. 2013]

Figure 6. Excerpt of artifacts generated for the first scenario

and activities. All the three techniques from this scenario (LSI, FCA, and clustering) were present in our tool, and PAxSPL also gave support when selecting them. Figure 6(b) presents part of the report detailing one activity instantiated into the process as well as one input and one output artifact. In this case, the *Divide features with LSI* activity is detailed. This activity contains two input artifacts, the *Object-oriented* source code and the *Features descriptions*; and one output artifact, the *common and variable partitions*. This report is important as it demonstrates how the activities were performed during the reengineering and the artifacts generated from them. This can be used as the requirements documentation of the SPL, also allowing evolution planning.

The second scenario was defined by extracting the information from [Acher et al. 2013]. In this case, the process was composed of four activities which were described in Table 1. Although it was possible to assemble all activities from the original study, we faced some challenges, similar to the first scenario. In comparison to the first scenario, this scenario uses different artifacts types, and two different techniques. The third technique, clustering, was also present in the first scenario, however, its application (an activity where it was applied) was different from scenario one. The third scenario used in the evaluation was extracted from [Al-Msie'Deen et al. 2012]. Considering the challenges faced when assembling this scenario, we faced two that were present in the previous scenarios. This scenario also has some similarities with those described earlier, as two retrieval techniques (one from the first and one from the second scenario) were also present here. We extracted the fourth scenario from [Shatnawi et al. 2014]. The FCA and dependency analysis techniques were present in this scenario either. However, one technique used in the original study was not supported by PAxSPL.

The fifth scenario used during the evaluation was extracted from [Alves et al. 2008]. Considering the similarities of the scenario, the clustering, and LSI techniques were also present here. Also, the Vector-Space Model technique was used, being the only scenario where it was present. The sixth scenario was extracted from [Chen et al. 2005] In this scenario, only the clustering technique was used, making it the most common retrieval technique up to this point. The seventh scenario was defined by extracting the information from [Paškevičius et al. 2012]. The clustering and dependency analysis techniques were present once again. This scenario also shares characteristics concerning the artifacts used, which were object-oriented source code,

the same as the first, third, and fourth scenarios. The eighth scenario was extracted from [Breivold et al. 2008]. This scenario also used the dependency analysis retrieval technique, which appeared in the other three scenarios. Another similarity is the artifacts type, in this scenario architecture descriptions and source code were used, similar to the second and fourth scenarios.

4.5. Answering the RQs

With the analysis of the evaluation results, we were able to answer our RQs, as discussed in the following.

RQ1. Is PAxSPL customizable to different scenarios? Although we faced different challenges in all eight scenarios where our framework was applied, the results pointed out the PAxSPL applicability in each individual scenario. In this sense, the PAxSPL supported 100% of the artifacts (see Table 2), 100% of activities (considering minor changes), and approximately 85% of the techniques (17 out of 20). As we replicated each original scenario as similar as possible, our main goal when analyzing this RQ was to understand the benefits of maintaining the aforementioned information. In this sense, we could see that such information is important for deciding which artifacts, techniques, and activities would be used, as well as for maintaining the reasons for making these decisions. This result is evidence by the reports generated in the tool, *e.g.*, Figures 6(b). These artifacts are important for the planning phase of PAxSPL because they aid the users in abstract the techniques selection process by making them (users) consider different aspects of their context. Also, the reasons for making the decisions are important to understand why the decisions worked (or did not work) when performing the reengineering. Therefore, we can also argue that our framework provides a way of maintaining a repository of artifacts related to the reengineering. These artifacts can be used when transforming the legacy system into an SPL, as well as when planning the SPL evolution.

RQ2. How does PAxSPL suit different scenarios? As presented in Table 1, all eight scenarios were different, using a variety of different artifacts, techniques, and activities. Table 2 summarizes these results, showing how many artifacts, activities, and techniques from the original studies were supported by PAxSPL. All artifacts used in the original studies could also be used in PAxSPL, which shows that concerning artifacts flexibility, our framework handled more than 20 different types of artifacts. The same results apply to the activities instantiated, as all activities from the original studies could be instantiated into the PAxSPL generic process. Also, five different retrieval techniques were used, and their different combinations made all scenarios unique. This variety among the scenarios, alongside the fact that we could instantiate all the processes, gives us evidence suggesting that our framework is indeed customizable for different scenarios. However, we faced a few challenges. Some of these challenges directly impact the results of this RQ. These challenges were not crucial problems, however, they must be considered aiming at improving our proposal. For instance, some techniques from the original studies were not supported by PAxSPL. Hence, these techniques could not be instantiated into the retrieval process. In addition, other challenges were identified during the evaluation. These challenges are discussed in the following section.

RQ3. What challenges are observed by customizing PAxSPL? Table 3 summarizes the eight challenges identified during the evaluation. Also, it presents possible solutions

Table 2. Results from the Evaluation.

Ref.	Art.		Tech.		Act.		Challenges							
	O	S	O	S	O	S	C1	C2	C3	C4	C5	C6	C7	C8
[Eyal-Salman et al. 2013]	6	6	4	4	5	5	✓	✓	✓					
[Acher et al. 2013]	7	7	3	2	4	4	✓	✓	✓	✓				
[Al-Msie'Deen et al. 2012]	5	5	3	3	3	3		✓	✓		✓	✓		
[Shatnawi et al. 2014]	5	5	3	2	4	4			✓	✓			✓	
[Alves et al. 2008]	4	4	3	2	3	3			✓		✓			
[Chen et al. 2005]	4	4	1	1	4	4		✓						✓
[Paškevičius et al. 2012]	7	7	2	2	6	6		✓						
[Breivold et al. 2008]	9	9	1	1	5	5		✓	✓				✓	

O - Original Study; S - PAxSPL Support.

Table 3. Challenges Identified During the Evaluation.

ID	Description	ID	Possible Solution
C1	Some activities from the original study were performed in parallel	PS1	Change the generic process to support parallelism
C2	An activity did not apply any retrieval technique	PS2	Allow the users to define activities without the application of retrieval techniques
C3	Some phases of the generic process did not have an activity related to it	PS3	Change the generic process to make activities or-optional
C4	A technique that was not supported by PAxSPL was used in an activity	PS4	Allow the user to add new techniques into the framework
C5	One activity used more than one retrieval technique	PS5	Allow the user to assemble multiple retrieval techniques into an activity
C6	One activity was part of more than one phase of the generic process	PS6	Allow the user to create activities that are part of more than one phase
C7	The approach did not aim at generating a FM	PS7	Change the last activity to aim at creating any kind of variability model
C8	The original scenario had no input artifacts	PS8	Make the inclusion of technical artifacts optional during the initiation step

to address them. The first challenge (C1) occurred in the first two scenarios. As the generic process does not support parallel activities, we had to handle this problem by transforming these activities into linear ones. However, a possible solution (PS1), would be to change the generic process and its instantiation to support parallel activities. The second challenge (C2) was the most frequent, appearing in six scenarios. This challenge occurred when an activity from the original study did not apply any retrieval technique. We argue that PS2 may address it by changing the framework to allow the users to assemble activities in the generic process without the need of assembling retrieval techniques for them. Challenge C3 was related to some phases of the generic process not receiving activities during the assembly. For instance, in the second scenario, the *group* activity was not used. As defined in its BPMN representation, however, all its phases are mandatory. Changing the generic process to make its activities Or-optional, meaning that at least one is mandatory is a possible solution (PS3). Challenge C4 was faced twice when techniques that were not present in the framework were used. We believe that C4 can be addressed by allowing the users to add or suggest the addition of new retrieval techniques into our framework (PS4). This could be done by providing a template or an online form for users to fill and send us.

Challenge C5 was identified when one of the activities from the original study applied more than one retrieval technique in parallel. As our framework does not give support to this, we should allow users to assemble multiple retrieval techniques into a single activity (PS5). The next challenge, C6, is related to one activity being part of more

than one phase of the generic process, for instance, extract and categorize. The possible solution (PS6) would be to allow the users to create activities that are part of more than one phase. The seventh challenge identified (C7) was faced when two of the scenarios did not aim at generating an FM at the end of their process. This challenge is different because PAxSPL's final output should be the FM. However, we still understand that C7 is an opportunity for improvement as a company that works or desires to have a variability artifact different from an FM, such as an SPL architecture. Thus, we plan to change the last activity of the process to support the creation of different variability models (PS7). Challenge C8 appeared only in one scenario [Chen et al. 2005], where their approach did not have input-only technical artifacts. Input-only artifacts are those collected and analyzed during the PAxSPL's initiation step, which can be organizational or technical. To address C8, we would make the inclusion of technical artifacts optional during the initiation step (PS8), relying on the decision only on organizational artifacts.

By analyzing these challenges, we identified at least eight points (possible solutions) for improving PAxSPL. However, we can also argue that these results can contribute to other researchers/practitioners that are developing or applying similar proposals/tools considering different scenarios. As most of the challenges faced were related to the differences across the scenarios used as input for the evaluation, such challenges can occur when using similar tools and proposals. Thus, the possible solutions identified can also be used to address these challenges.

4.6. Threats to Validity

Next, we present the threats of our study and how we tried to mitigate them [Wohlin et al. 2012].

Construct validity: an important threat concerns the selection of relevant scenarios for conducting the customization. This may be a threat if the scenarios selected were not reliable enough to be considered relevant when extracting and analyzing the results of the evaluation. To mitigate this, we used a well-known catalog of SPL case studies [Martinez et al. 2017]. Another threat is related to the selection of scenarios that may or may not be useful for our evaluation, as they may not possess the information we require. To mitigate this problem, we defined and applied three inclusion criteria when selecting the studies from where the scenarios would be extracted.

Internal Validity: a possible threat is related to errors when conducting and documenting the results of the evaluation. This may lead to erroneous conclusions, resulting in misleading answers for the RQs. To mitigate this problem, we clearly defined four steps for conducting the evaluation. Also, we conducted and published a pilot execution [Marchezan et al. 2020] to evaluate how the steps performed helped to answer the RQ.

External Validity: concerning the relevance of our findings to other studies, we established a replicable protocol, using as input studies found in a public and well-known catalog of case studies. Also, we provide all artifacts from the evaluation in an open data repository [6], allowing reproducibility.

Reliability: a possible threat is related to problems with the data dependency caused by the researchers conducting the evaluation. To mitigate this problem, we defined four

different metrics for the RQs, describing how they may be answered by analyzing the results. An additional threat is the number of scenarios selected. Despite having only eight scenarios, we argue that they allowed us properly investigating the flexibility and limitations. However, we understand that a larger dataset could complement our results.

5. Conclusion

The SPL reengineering process is adopted by organizations that aim to migrate their legacy software into a technology focusing on systematic reuse. Organizational scenarios impact the SPL reengineering process as scenario aspects can determine which retrieval techniques are more suited to them. In this sense, a process customized based on these aspects may be a satisfactory solution for reducing this impact.

Therefore, we propose a framework called PAxSPL that gives support for users to plan the SPL reengineering by customizing their feature retrieval processes. Our framework was constructed by considering the SPL reengineering planning activities, limitations of works in the field, and requirements of end-user tools. As the results of our evaluation pointed out, PAxSPL can be applied in different scenarios considering a variety of scenario variables such as artifacts, techniques, and activities. For future work, we intend to extend PAxSPL to include additional feature retrieval techniques found in the literature. In addition, we plan to interview the authors of the eight scenarios used in the evaluation to collect and analyze their opinion concerning the instantiation of the scenarios.

References

Acher, M., Cleve, A., Collet, P., Merle, P., Duchien, L., and Lahire, P. (2013). Extraction and evolution of architectural variability models in plugin-based systems. *Software & Systems Modeling*, 13(4):1367–1394.

Al-Msie'Deen, R., Seriai, D., Huchard, M., Urtado, C., Vauttier, S., and Eyal-Salman, H. (2012). An approach to recover feature models from object-oriented source code. *Actes de la Journée Lignes de Produits*, pages 15–26.

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

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

Breivold, H. P., Larsson, S., and Land, R. (2008). Migrating industrial systems towards software product lines: Experiences and observations through case studies. In *2008 34th Euromicro Conference SEAA*, pages 232–239.

Capilla, R., Gallina, B., Cetina, C., and Favaro, J. (2019). Opportunities for software reuse in an uncertain world: From past to emerging trends. *Journal of Software: Evolution and Process*, 31(8):e2217.

Chen, K., Zhang, W., Zhao, H., and Mei, H. (2005). An approach to constructing feature models based on requirements clustering. In *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*, pages 31–40. IEEE.

Clements, P. and Northrop, L. (2002). *Software product lines*. Addison-Wesley,.

Eyal-Salman, H., Seriai, D., and Dony, C. (2013). Feature-to-code traceability in a collection of software variants: Combining formal concept analysis and information retrieval. In *14th IRI*, pages 209–216. IEEE.

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

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

Krüger, J., Mahmood, W., and Berger, T. (2020). Promote-pl: A round-trip engineering process model for adopting and evolving product lines. In *24th ACM Conference on Systems and Software Product Line: Volume A*, SPLC '20, New York, NY, USA. ACM.

Laguna, M. A. and Crespo, Y. (2013). A systematic mapping study on software product line evolution: From legacy system reengineering to product line refactoring. *Science of Computer Programming*, 78(8):1010–1034.

Marchezan, L. (2020). PAxSPL: A generic framework to support the planning of SPL reengineering. Master's thesis, Universidade Federal do Pampa, Av. Tiaraju, 810 - Ibirapuitã, Alegrete - RS, 97546-550, Brazil.

Marchezan, L., Carbonell, J. a., Rodrigues, E., Bernardino, M., Basso, F. P., and Assunção, W. K. G. (2020). Enhancing the feature retrieval process with scoping and tool support: Paxspl_v2. In *Proceedings of the 24th ACM SPLC - Volume B*, SPLC '20, page 29–36, New York, NY, USA. Association for Computing Machinery.

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

Martinez, J., Assunção, W. K. G., and Ziadi, T. (2017). Espla: A catalog of extractive spl adoption case studies. In *Proceedings of the 21st SPLC - Volume B*, SPLC '17, pages 38–41, New York, NY, USA. ACM.

Paškevičius, P., Damaševičius, R., Karčiauskas, E., and Marcinkevičius, R. (2012). Automatic extraction of features and generation of feature models from java programs. *Information Technology And Control*, 41(4):376–384.

Pereira, J. A., Constantino, K., and Figueiredo, E. (2015). A systematic literature review of software product line management tools. In *International Conference on Software Reuse*, pages 73–89. Springer.

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

Shatnawi, A., Seriai, A., and Sahraoui, H. (2014). Recovering architectural variability of a family of product variants. In Schaefer, I. and Stamelos, I., editors, *Software Reuse for Dynamic Systems in the Cloud and Beyond*, pages 17–33, Cham. Springer International Publishing.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*, volume 1. Springer Science & Business Media.