Enhancing the Feature Retrieval Process with Scoping and Tool Support – PAxSPL_v2

Luciano Marchezan
lucianomarchp@gmail.com
Federal University of Pampa
Alegrete, RS

João Carbonell
joaocarbonellpc@gmail.com
Federal University of Pampa
Alegrete, RS

Elder Rodrigues
eldermr@gmail.com
Federal University of Pampa
Alegrete, RS

Maicon Bernardino
bernardino@acm.org
Federal University of Pampa
Alegrete, RS

Fábio Paulo Basso
fabiopbasso@gmail.com
Federal University of Pampa
Alegrete, RS

Wesley K. G. Assunção
wesleyk@utfpr.edu.br
Federal University of Technology
Toledo, PR

ABSTRACT

Software Product Lines (SPLs) are commonly adopted with an extractive approach, by performing a reengineering process in legacy systems, when dealing with variability and reuse became challenging. As a starting activity of the process, the legacy systems are analyzed to retrieve, categorize, and group their features in terms of commonality and variability. Due to the importance of this feature retrieving, we proposed the Prepare, Assemble, and Execute framework for SPL reengineering (PAxSPL). PAxSPL aims at guiding users to customize the feature retrieval for their scenario. In an initial evaluation of the PAxSPL in a real-world scenario, we could observe the need for including scoping activities and implementing a tool to make the framework more adoptable in practice. In this paper, we describe how we performed these improvements. We performed the evolution of PAxSPL by including SPL scoping concepts and activities into our framework as well as developing a supporting tool. We also conducted a pilot study to evaluate how PAxSPL allows instantiating a scenario where the SPL reengineering were conducted. The results show that all artifacts, activities, and techniques from the scenario could be properly represented. However, we also identified a potential limitation during the assembly of techniques regarding parallel activities. The main contribution is PAxSPL_v2 that makes the framework more adherent to industries performing the reengineering of legacy systems into SPLs.

CCS CONCEPTS

• **Software and its engineering → Software product lines**.

KEYWORDS

software product lines, variability management, automated support

ACM Reference Format:
Luciano Marchezan, João Carbonell, Elder Rodrigues, Maicon Bernardino, Fábio Paulo Basso, and Wesley K. G. Assunção. 2020. Enhancing the Feature Retrieval Process with Scoping and Tool Support – PAxSPL_v2. In . ACM, New York, NY, USA, 8 pages.

WEESR 2020, October 19–23, 2020, Montreal, CA

1 INTRODUCTION

The increasing demand for high-quality software and the concern of software development companies for lowering project costs has led to the emergence of software engineering solutions such as the Software Product Line Engineering (SPLE) [22]. SPLE is a reuse-oriented approach to develop Software Product Lines (SPLs), which are families of product variants, allowing the cost-efficiently derivation of tailored products to specific markets, utilizing common and variable assets in a planned manner [7]. The benefits and motivation for adopting SPL are manifold, such as reduced maintenance effort, reduced time-to-market, increased reusability, to cite some [19]. Among the approaches to adopt SPLs, the most common in practice is the extractive one, where existing systems variants are the starting point for all SPLE activities [4, 12].

The extractive adoption of SPLs is performed by the conduction of a reengineering process, in which legacy assets are transformed into SPL assets and reused/customized to systematically create software products [4]. The reengineering process can be seen as the process of evolving legacy systems into SPLs by identifying and extracting common and variable assets [13]. The reengineering process basically uses a number of product variants as input, which are analyzed and transformed into an SPL as output. In the literature, we can find a wide range of techniques, methods, and some tools to support this process [4]. In the scope of our work, the analysis of the legacy systems, refereed from now on as *feature retrieval*, comprehends the retrieving, categorization, grouping, and organization of features, which are the building blocks of SPLs [11].

Due to the paramount importance of the reengineering process for extractive adoption of SPLs, in previous work, we proposed the Prepare and Assembled framework (PAxSPL) [14]. PAxSPL is a framework supported by guidelines that help users to customize the feature retrieval process for the context of their companies. After the execution of a case study for evaluating PAxSPL, we identified the need for improvements for better adherence to industrial contexts. Among the conclusions of the case study, we could observe that activities covering SPL scoping were not given the required focus as part of PAxSPL. SPL scoping would help engineers handle more business-related aspects of the reengineering process. Also, the case study results pointed out evidences that some PAxSPL activities required high effort due to the need for keeping artifacts

and reports updated and well-structured. This issue implied a need for a supporting tool to deal with this limitation.

The limitations aforementioned led us to evolve PAxSPL into a new version. In this paper, we present PAxSPL_v2, describing its characteristics and improvements in comparison to PAxSPL_v1. In addition, we present a pilot study conducted to analyze how PAxSPL_v2 can be instantiated in a scenario extracted from the literature that focused in the reengineering process. The results indicate that PAxSPL_v2 can be properly instantiated using different artifacts, activities, and techniques. However, one limitation was identified with regard to the assembly phase, which does not allow the assembly of parallel activities.

The contributions of our work are: (i) evolution of the PAxSPL to include SPL scoping activities, allowing SPL scoping customization for specific contexts; (ii) development of a supporting tool that can be used for executing all PAxSPL activities; and (iii) empirical results of a pilot study to collect evidence about how PAxSPL may be instantiated to a real scenario, and to identify possible limitations.

This work is structured as follows: Section 2 presents PAxSPL_v1 and the limitations identified in a case study. Section 3 presents the improved PAxSPL framework. Section 4 presents details of a pilot evaluation, discussing the preliminary results. Section 5 discusses related works. Lastly, Section 6 presents concluding remarks.

2 BACKGROUND

In this section, we describe the first version of our framework (PAxSPL_v1) and the results of its evaluation, which revealed the need for improvements, motivating the work described in this paper.

2.1 PAxSPL_v1

The Prepare, Assemble and Execute framework (PAxSPL) [14] was proposed to support feature retrieval for SPL reengineering. This support is done by aiding the decisions related to the selection of strategies and techniques for feature retrieval. To this end, we defined the main workflow of PAxSPL, as presented in Figure 1: (i) *prepare*, on the left side of the figure; (ii) *assemble*, in the middle; and (iii) *execute*, on the right. Next, these steps are described in detail.

Prepare: This step is responsible for collecting the information needed for the next steps of PAxSPL. In the first activity, information

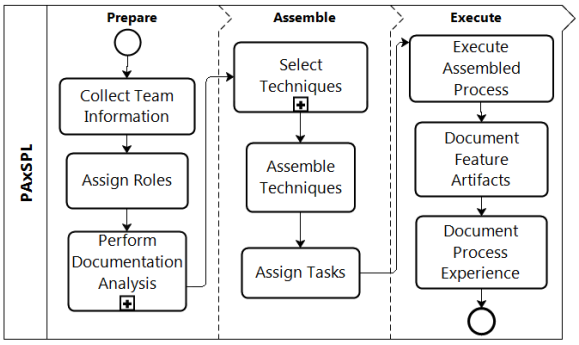

Figure 1: PAxSPL_v1 workflow.

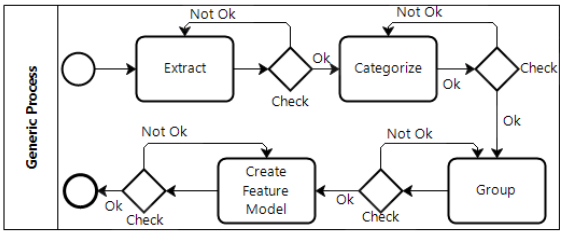

Figure 2: The Generic Process for Feature Retrieval.

about the team is collected, including the experience and skills. The second activity is *Assign Roles* based on the information collected on the previous activity. Then, a documentation analysis is performed on product architecture, requirements, domain information, and team information.

Assemble: The first activity is to *Select Techniques*, in which the information collected previously is analyzed to help the selection of techniques for the feature retrieval. The second activity during the Assemble phase is *Assemble Techniques*, where the chosen techniques are assembled into our generic process, as presented in Figure 2. The generic process consists of the main activities performed during the feature retrieval, observed in the literature when we were defining PAxSPL. *Extract*, *Categorize* and *Group* are basic activities performed with the features prior to the creation of the feature model. The possible retrieval techniques are presented in Figure 3. The assembled process is customized according to the scenario where PAxSPL is being applied, giving our process flexibility to be applied in different situations. The last activity of *Assemble* is *Assign Tasks* where each member of the team will receive a task to perform during the retrieval process execution.

Execute: During this step, 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*, in which the artifacts are documented in a structured way, according to the techniques selected. Artifacts may be variability reports, feature descriptions, data dictionary, to cite some. 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 client demands or for different software products of the same organization, reducing cost and effort.

2.2 Retrieval Techniques and Case Study

As illustrated in Figure 3, 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, and Information Retrieval. For Static Analysis, we have three Or-alternatives: Dependency Analysis, Data-Flow analysis, and Clustering. For Information Retrieval we also have three Or-alternatives: Latent Semantic Indexing (LSI), Vector Space Model (VSM), and Formal Concept Analysis (FCA).

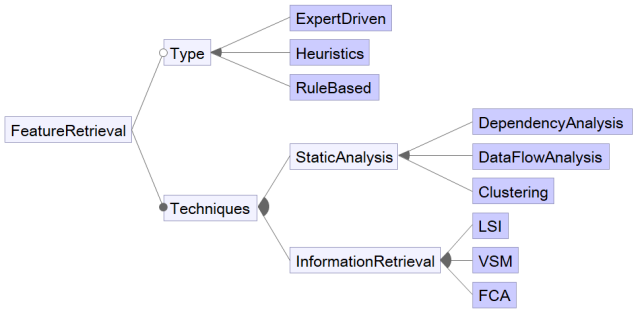

Figure 3: A Feature Model of Retrieval Techniques.

The second group is optional, composed of three techniques: Expert Driven Extraction, Rule-Based, and Heuristics. Based on the selection of the techniques, the user would assemble them into the generic process for feature retrieval and analysis, shown in Figure 2. As stated, the generic process is used to tailor the assembled process that generates the extracted features and the feature model [11].

In Figure 4, we present the process assembled during the case study to evaluate PAxSPL_v1 [14]. In this process, FCA and clustering were assembled as extraction techniques to retrieve the features and create the feature model. In this particular case, the tasks were not assigned to a specific actor, which means they may be performed by anyone. As Figure 4 presents, a concept lattice was checked to find problems. Then, the extracted features were grouped into clusters, and checked for inconsistencies was performed. The clusters were refined, the features were categorized and validated with the domain specialist. Lastly, the feature model was created and checked, and the process ended.

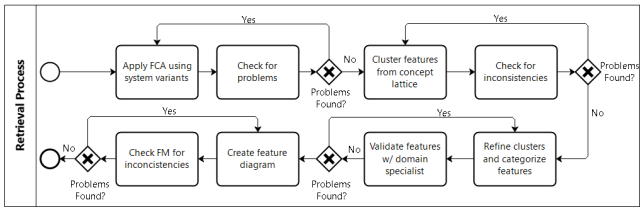

Figure 4: Case Study Assembled Process

2.3 Limitations Identified in PAxSPL_v1

A case study was conducted to evaluate how PAxSPL aids feature retrieval in the reengineering of system variants into an SPL. We collected evidence about the relevance of PAxSPL artifacts, effort to apply our framework, and its reuse capability when reapplied in a different scenario [14].

By analyzing the results, we have identified the needs for improving PAxSPL. One limitation was the lack of SPL scoping related activities when conducting the SPL reengineering. The inclusion of these activities would allow handling more business-related artifacts, supporting the decision-making when identifying, extracting, and analyzing the features. Another opportunity for improvement was the high effort of some activities when compared to others. For instance, an activity called *Execute Assembled Process* required more than twice the effort of the second most demanding activity. Another problem identified was related to the need of keeping updated, organized, and well-structured the artifacts generated during the reengineering, *e.g.*, reports and documentation. This proved to be an issue as in the case study, as the participants spent a considerable amount of time organizing artifacts. Approaching the participants of the case study, we could observe that these two issues could be mitigated with tool support to help the management and execution of the process. This observation corroborates findings previous studies, that pointed out the lack of tools to support SPLE [4, 18].

Based on observed limitations, we evolved the PAxSPL framework, defining PAxSPL_v2, which is presented in the next section.

3 PAXSPL_V2

PAxSPL_v2 resulted from the evolution of our previous framework. In addition to the activities of PAxSPL_v1, this new version includes scoping activities and a tool support to automate the framework execution. These improvements are described in this section.

3.1 SPL Scoping Activities

The new version of the process follows the same structure of PAxSPL_v1, however, we included a lane pool for SPL scoping activities, as presented in Figure 5. Considering this lane pool, we have a parallel gateway which split the main workflow into the feature retrieval and scoping. Thus, still during the Assemble phase, the scoping process should be assembled using the scoping concept map (see Figure 6) and the scoping generic process (see Figure 7). In parallel, while scoping process is executed (*Execute Scoping Process*), scoping artifacts may be traced to feature artifacts.

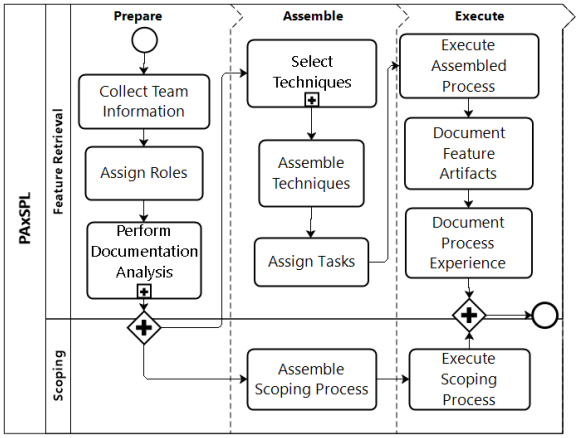

Figure 5: PAxSPL_v2 including SPL Scoping Activities.

In addition to the feature retrieval, the new process also guides customization considering the scope of the SPL. By analyzing a set of 45 studies describing SPL scoping methods and strategies, we were also able to establish a feature model of scoping concepts. Figure 6 presents these concepts which are divided by scoping type and supporting concepts. All features in the model are Or-alternative,

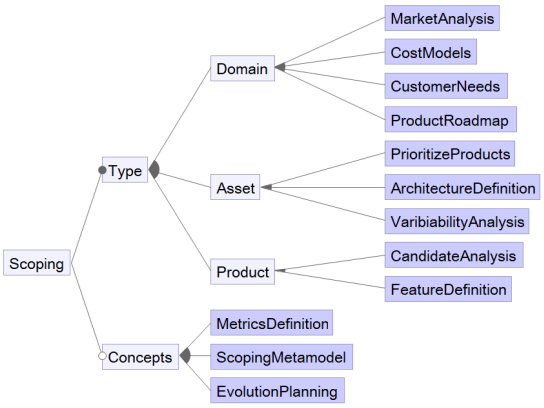

Figure 6: A Feature Model of SPL Scoping Concepts

except for the supporting concepts, which are optional. Among the concepts, we have the definition of metrics for scoping, use or definition of meta-models, and the SPL evolution plan. Regarding the scoping types, they are: Domain, Asset, and Product. Domain is subdivided into market analysis, cost analysis (and its variations), and product roadmap. Continuing, the asset scoping may be divided into prioritize products, the definition of architecture, and variability analysis. Lastly, product scoping, also called product portfolio, is composed of candidate analysis and feature definition.

Similar to the feature retrieval strategies, the scoping activities and concepts must be selected by PAxSPL_v2 users and assembled into a generic SPL scoping process. Figure 7 presents the generic SPL scoping process to be used as basis for the assembling/customization. *Pre-Scoping* is the first activity, where supporting concepts from the scoping feature model may be used, such as metrics definition.

The users would then assemble the activities related to which scoping type. The selection of activities is performed based on the user's context. As presented in the BPMN model (Figure 7), the activities related to the scoping type are not mandatory, however, at least one must be performed. Lastly, the *scoping closure* activity is executed. This activity is generic as several ways to close the scoping process may be performed. To better represent this process, we present an example of the scoping process assembled in Figure 8. The process starts by defining metrics based on business aspects, then deriving these metrics for specific projects. Both of these activities are part of the pre-scoping. After the pre-scoping is finished, the candidate features are analyzed and their names and descriptions are defined, these two activities are examples of *product scoping* activity. While these are executed, the market is analyzed in an activity related to *domain scoping*. Lastly, a plan for evolving the SPL is defined in the *scoping closure* activity.

3.2 Supporting Tool

In this section, we present the supporting tool[1] developed to aid the execution of the PAxSPL framework. We defined that the tool

[1]A tool demo video is available at https://youtu.be/VjxCshdMExk

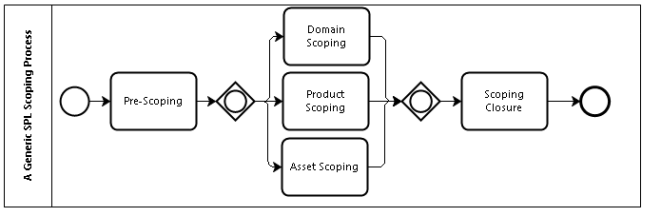

Figure 7: Generic SPL Scoping Process.

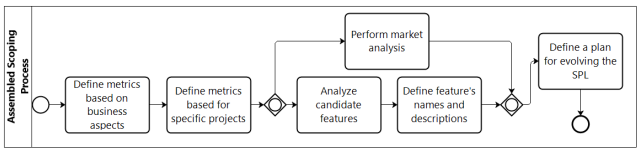

Figure 8: An Assembled SPL Scoping Process.

would have four distinct actors, and nine use cases (UC). The actor *Manager* performs the use cases (UC1) Manage Project, (UC2) Manage Team, and (UC3) Document Process Experience. The actor *Feature Retriever* performs (UC4) Execute Retrieval Process. *Feature Tester* performs (UC5) Check Features Artifacts. Lastly, *Team Member* performs (UC6) Select Strategies and Techniques, (UC7) Generate Documentation Set, (UC8) Create Feature Model, and (UC9) Configure Products.

After deriving these use cases into more detailed requirements, we designed and implemented our tool. The tool supports the execution of all activities of the process life cycle (Figure 5), as well as the activities assembled into the generic processes (Figures 2 and 7). For instance, for the `Collect Team Information` activity, the tool allows users to include team members and fill out their company role, experience, and knowledge about feature retrieval techniques.

As described by the use cases, the tool aims at reducing the effort when conducting the feature retrieval and scoping process. This reduction is performed, mainly, by structuring the way information about the SPL reengineering is registered in the tool. Also, by allowing multiple users to work in parallel in different activities, the retrieval process may be performed more efficiently. An example of functionality that aims at reducing the effort it presented in Figure 9. In this screen, the tool suggests feature retrieval techniques may be used in the current scenario. To calculate this suggestion, the tool analyzes the profile of the team and the artifacts available, both registered by the users. For instance, if a team member has experience with FCA and if an artifact registered may be used by this technique, the recommendation level of the techniques is higher.

Another aspect of the tool that intends to reduce the effort is the generation of reports. In the case study previously discussed [14], participants spent a considerable amount of time filling and updating the reports during activities. By implementing this functionality in the tool, all reports are now generated and updated automatically based on the information saved in the tool's database.

An important part of the tool is feature analysis. Users can register the features retrieved, their descriptions, relations, and types. Also, the artifacts (registered previously) may be linked with the

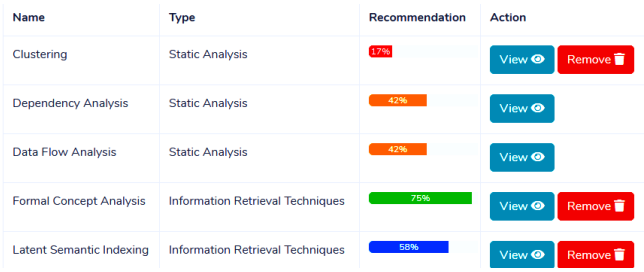

Figure 9: Feature Retrieval Techniques Screen

features, generating a traceability matrix. Another functionality of the tool is to allow users to configure/derive products from the feature model. Figure 10 shows the configuration screen, where the features may be selected based on the rules of the feature model. The product has a name and a description. After configuring a valid product, the configuration can be downloaded as an XML file. This XML file can be imported in the FeatureIDE[2]. Additionally, reports of the features and artifacts present in each product may be downloaded, and it is possible to visualize a matrix showing the features that are present in each product.

4 PILOT STUDY

Next we describe a pilot evaluation of PAxSPL_v2. This evaluation aims to collect preliminary results of our improved framework and supporting tool.

[2]https://featureide.github.io/

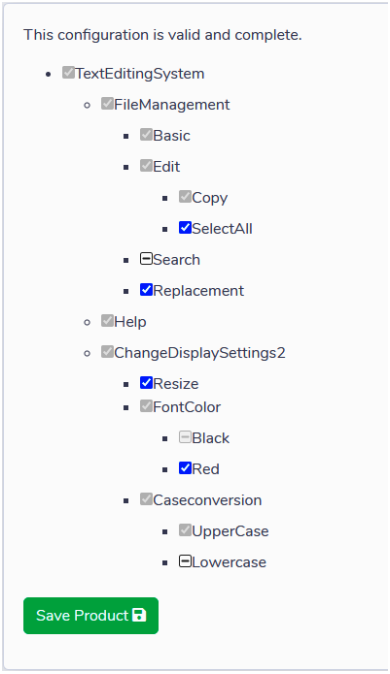

Figure 10: Product Configurator Screen

4.1 Evaluation Design

The goal of the pilot study is to evaluate how PAxSPL_v2 can be instantiated for an existing scenario in order to conduct SPL scoping and using the supporting tool. Thus, we aim at customizing PAxSPL_v2 to different scenarios observed in related works, which will serve as ground truth for our analysis. The Research Questions (RQs) that guided the evaluation are:

RQ1. Can PAxSPL_v2 be instantiated to represent the activities, artifacts, and techniques used in scenarios were SPL reengineering was conducted? This RQ aims to evaluate how our framework can be instantiated to be applied an existing scenario, addressing activities, artifacts and techniques for SPL reengineering.

RQ2. What limitations are observed by customizing PAxSPL_v2? We aim at identifying and understand which issues and limitations were faced on adapting PAxSPL.

By answering these RQs, we want to obtain enough evidences for achieving our goal. In this sense, **RQ1** gives evidence about the adherence of PAxSPL in terms of customization. We also look at how the generic process would be instantiated. 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, **RQ2** was defined for collecting and analyzing these challenges and how they have impacted the instantiation process, pointing out PAxSPL limitations.

4.2 Data Set

The data set used as input for this evaluation is composed of studies where SPL reengineering was applied. More specifically, we selected studies from the ESPLA catalog [15], which is a collaborative catalog of case studies on SPL reengineering. To selected the subject studies, we applied three criteria. The study should satisfies all criteria to be considered for this evaluation. These criteria are:

i **The study must apply at least one retrieval technique present in the PAxSPL guidelines**. Hence, we only considered studies which used retrieval techniques also covered by PAxSPL;

ii **Each new study must present a scenario different from other approaches already selected**. In this incremental selection process we want to guarantee that all selected studies presented different scenarios from one another. We considered different scenarios when the study: (i) used at least one different retrieval technique; (ii) used at least one different input artifact; or (iii) had a different workflow when applying the feature retrieval techniques.

iii **The study evaluation protocol, data set, and results must be available online**. This allows us to use the artifacts to evaluate our framework and enables replications.

4.3 Procedure

During the execution of the pilot study, PAxSPL_v2 was applied in the scenario extracted from one study selected. In this case, we defined as a scenario: using the same inputs artifacts, and retrieval

techniques present in the selected study. Therefore, the team information used for our evaluation may be different from the original study, as the studies do not present this. Hence, once we started PAxSPL_v2 execution for one scenario, we intend to observe if by executing the tool, we would be able to assemble a retrieval process that would be fully executable in that scenario. Thus, we do not intend to compare the quality or precision of the features extracted from legacy software using our framework. We are limiting the evaluation at only analyzing whether PAxSPL_v2 can represent the feature retrieval process similarly to the original process from the study. More specifically, when conducting the evaluation, 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;
iv Use in PAxSPL_v2, with the tool supporting, the artifacts, techniques, and activities identified.

After conducting these steps, we answered the RQs by analyzing the following metrics. For **RQ1**, (i) the number of artifacts from the original study that were used by PAxSPL_v2; (ii) the retrieval techniques that were assembled into PAxSPL generic process; and (iii) the activities that were assembled into our framework. For **RQ2**, we collected any limitation faced during execution of the study.

4.4 Execution

For assessing our evaluation protocol, we performed a pilot execution randomly selecting one study, and executing PAxSPL_v2 in its scenario. The study selected was [9], as it satisfied our selecting criteria. Table 1 presents the data identified prior to executing PAxSPL_v2. The authors used two input artifacts, namely object-oriented source code and feature descriptions. LSI and FCA were applied for feature retrieval among five different activities. One of these activities is the derivation of *code-topics*, which are artifacts used for connecting features with source code. Hence, these *code-topics* are categorized as clusters containing the similar classes (Java source-code). Thus, we considered that clustering was also used by this work. Another important point is that their process does not have activities regarding SPL scoping. We executed PAxSPL_v2 using the supporting tool. The following section presents the artifacts generated during the execution.

Table 1: Data from the Original Studies

Ref.	Artifacts	Tech.	Activities
[9]	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.

4.5 Pilot Study Results

After extracting the data from the original study, we were able to assemble and execute their process into the PAxSPL tool. In the following, we present some artifacts generated (by our tool) during this execution[3].

Figure 11 presents the BPMN representation of the assembled retrieval process. This representation is the result of assembling the techniques selected in the generic process. The five activities and their workflow were based on the original study, however, minor modifications were made, such as writing the first words as verbs.

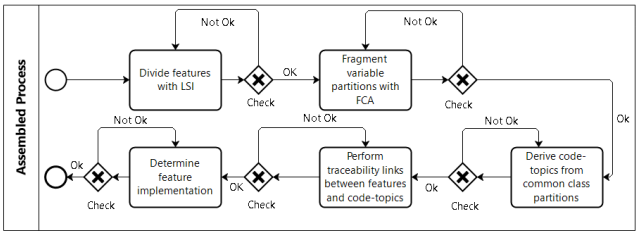

Figure 11: BPMN Process Assembled for [9]

Figure 12 presents part of the report detailing the assembled process. The report shows all the five activities assembled into the process as well as their input and output artifacts. In this case, the *Divide features with LSI* activity are 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*.

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:2	**Input**	
		Name:	**Feature Description**
		Type	Domain
		Description:	Description of features of the argoUML system.
		Extension:	.doc
		Link (url):	https://github.com/argouml-tigris-org/argouml
		Last Update:	05-24-2020 by Luciano
	Artifact:3	**Output**	
		Name:	**Common and variable partitions**
		Type	Development
		Description:	Classes that implement common and optional features
		Extension:	.lsi
		Link (url):	https://github.com/argouml-tigris-org/argouml
		Last Update:	05-27-2020 by Luciano

Figure 12: First Part of the Process Assembled Report for [9]

[3]Artifacts available at https://github.com/HestiaProject/PAxSPL/tree/master/process/evaluation

4.6 Preliminary Considerations

As this is a pilot study, we only have preliminary results for answering our RQs. **Answering RQ1**: we were able to include all input artifacts and use them during the execution. Also, all activities from the original work were assembled into our generic process. This may indicate that PAxSPL_v2 was accordingly instantiated to the scenario. However, we should include new scenarios to collect further evidence to support this claim. Also, as we could execute the assembled process without any major problems, PAxSPL_v2 was properly instantiated in ways that are well-adapted to the scenario and the user experience. However, we also identified that the generic feature retrieval process may have to be changed. This change was related to the workflow of the generic process, which consists of four main activities executed in sequence: extract, group, categorize, and create feature model. When assembling the process from [9], we noticed that some activities from the original study were performed in parallel, *e.g.*, activities 1 and 2. **Answering RQ2**: as our generic process does not handle this parallelism, we were forced to modify the original workflow to be executed as a straight process. Before performing changes in the generic process, however, we also intend to perform the evaluation with other cases. This issue also qualifies as a potential challenge. **Summary**: when considering the evaluation protocol, we believe that it satisfies our needs for collecting the evidences required for answering our RQs. This is due to the selected study being suited for our evaluation, as well as the analysis of the artifacts generated which provided some evidence about possible aspects to be improved in PAxSPL.

4.7 Threats to Validity

Next we present the threats of our study and how we tried to mitigate them [20, 24].

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 [15]. Yet, the catalog is composed of studies from the literature, therefore, there is also the threat that these scenarios might not represent real world-scenarios. 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. Taking into account our pilot evaluation, the threat might be the use of only one scenario to obtain the preliminary results. However, we selected a scenario that well-represents the context in which our study is inserted.

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 the pilot execution to evaluate how the steps performed helped to answer the RQ.

External Validity: concerning the relevance of our findings to other studies in the field, 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 repository, 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.

5 RELATED WORK

In this section, we describe related work based on their characteristics and discuss how they may be compared to PAxSPL_v2.

5.1 Studies on Feature Retrieval

Considering the artifacts classification, requirements artifacts are used in most studies [1, 6, 16], becoming the most common artifact amongst the approaches. Domain information is also used [2], being relevant in the SPL context. Source code is used in different proposals [10, 16]. Despite these approaches using different types of artifacts, their flexibility could still be improved. For instance, in [6] all artifacts are mandatory, reducing the customization according to the user's scenario. The proposal presented in [16], however, was designed to be generic regarding the artifact types, because it gives support for integrating new artifact types. With regard to the strategies used for feature retrieval, there is also a lack of flexibility. Despite some studies using more than one strategy type [1, 16], these strategies are mandatory, similar to the lack of flexibility concerning artifacts. Once again, [16] gives more flexibility to the user because it allows new strategies to be integrated by extending their proposal. Also, some studies use retrieval techniques that are not present in PAxSPL guidelines. Acher et al. [1] use structural similarity to identify architecture FM from plugins. In [16], authors use overlaps for identifying elements that compose an artifact, which will later be analyzed to extract features. Lastly, in [2] propositional logic is applied for extracting FM from product descriptions.

5.2 Guidelines to Support SPL Scoping

Guidelines to support SPL scoping are present in studies such as [17], and PLEvo-Scoping [23]. However, in comparison to PAxSPL, such guidelines fails on providing information such as the recommendation scenarios. PAxSPL supports users with low experience regarding SPL scoping, which are not the goal of the guidelines presented by the aforementioned studies. These studies use Thinklets to describe aspects of the scoping process, such as collaborative and prioritization techniques, used for defining a product roadmap. In [23], on the other hand, authors present the use of guidelines alongside procedural descriptions, checklists, and document templates to support the scoping team during the 13 activities of their process. This guideline is similar to those presented in PAxSPL.

5.3 Customization for Different Scenarios

The customization considering different scenarios and contexts is given by approaches such as PuLSE-Eco [21]. PuLSE-ECO is part of the PuLSE framework [8], focusing on SPL scoping. As part of PuLSE, the PulSE-Eco presents the customization of its components. Despite providing this customization, however, the PulSE framework does not provide some customization mechanisms as PAxSPL

does. PAxSPL offers a generic scoping process and the feature model of scoping concepts. Besides PuLSE-Eco, the work presented in [3] defines a set of rules for different scenarios. Their approach handles scoping costs according to the current scenario, thus, making their approach customizable. In contrast, when defining which scoping aspects to be used in PAxSPL, the users may also handle different aspects of scoping, including costs.

5.4 Tools

Although we may find different tools to support feature retrieval and scoping, the PAxSPL_v2 supporting tool is different from the existing ones by focusing on organizational aspects of the reengineering, *i.e.,* collecting and analyzing team information) instead of only technical information, which is the focus of other tools [5, 21].

6 CONCLUSION

Due to the limitations identified in a case study [14], namely scoping activities and supporting tool, we presented improvements to our SPL reengineering framework, originating PAxSPL_v2.

By including SPL scoping concepts and specific activities, we extended PAxSPL's life-cycle. We presented these improvements concerning SPL scoping, including an FM of SPL scoping concepts and a generic SPL scoping process that may be instantiated by assembling the concepts from the FM. We also developed a web-based supporting tool, which supports the execution of all activities. We discussed and presented this tool developed with the intention of reducing the effort of executing the framework.

With the goal to study PAxSPL_v2 instantiation capabilities and identify possible limitations, we defined a protocol for empirically evaluating our framework. The protocol, composed of two RQs, aims at collecting evidence of how PAxSPL_v2 may be instantiated based on scenarios extracted from the literature. To extract the scenarios, we applied three inclusion criteria in studies present in a catalog of SPL case studies [15]. For the evaluation of PAxSPL_v2, we conducted a pilot study. In this pilot study, we were able to apply our framework in a scenario extracted from [9]. The results pointed out that PAxSPL_v2 is instantiated addressing different artifacts, techniques, and activities. However, the results also evidenced a limitation related to the PAxSPL generic process, which does not support the assembly of activities performed in parallel. We plan to extend the results of the evaluation by including more scenarios extracted from the catalog, as well as identifying the limitations and planning improvements to address them.

ACKNOWLEDGMENTS

This work was partially funded by Unipampa 10/2019 - PAPG/2019, the Brazilian National Council for Scientific and Technological Development (CNPq) grant no. 408356/2018-9, and Araucária Foundation for Scientific and Technological Development of the State of Paraná (FAPPR) grant no. 51435.

REFERENCES

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

[2] Mathieu Acher, Anthony Cleve, Gilles Perrouin, Patrick Heymans, Charles Vanbeneden, Philippe Collet, and Philippe Lahire. 2012. On Extracting Feature Models from Product Descriptions. In *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems* (Leipzig, Germany) *(VaMoS '12)*. ACM, New York, NY, USA, 45–54. https://doi.org/10.1145/2110147.2110153

[3] Hamad I. Alsawalqah, Sungwon Kang, and Jihyun Lee. 2014. A method to optimize the scope of a software product platform based on end-user features. *Journal of Systems and Software* 98 (2014), 79 – 106. https://doi.org/10.1016/j.jss.2014.08.034

[4] W. Assunção, R. Lopez-Herrejon, L. Linsbauer, S. Vergilio, and A. Egyed. 2017. Reengineering legacy applications into software product lines: a systematic mapping. *Empirical Software Engineering* 22, 6 (2017), 1–45.

[5] Colin Atkinson, Joachim Bayer, and Dirk Muthig. 2000. *Component-Based Product Line Development: The KobrA Approach.* Springer US, Boston, MA, 289–309. https://doi.org/10.1007/978-1-4615-4339-8_16

[6] Guillaume Bécan, Mathieu Acher, Benoit Baudry, and Sana Ben Nasr. 2013. *Breathing Ontological Knowledge Into Feature Model Management.* Technical Report RT-0441. 15 pages. https://hal.inria.fr/hal-00874867

[7] Paul Clements and Linda Northrop. 2001. *Software Product Lines: Practices and Patterns.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[8] J. deBaud and K. Schmid. 1999. A systematic approach to derive the scope of software product lines. In *Proceedings of the 1999 International Conference on Software Engineering (IEEE Cat. No.99CB37002)*. 34–43. https://doi.org/10.1145/302405.302409

[9] H. Eyal-Salman, D. Seriai, and C. Dony. 2013. 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*. IEEE, 209–216.

[10] Stefan Fischer, Lukas Linsbauer, Roberto Erick Lopez-Herrejon, and Alexander Egyed. 2014. Enhancing clone-and-own with systematic reuse for developing software variants. In *Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on*. IEEE, 391–400.

[11] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. 1990. *Feature-Oriented Domain Analysis (FODA) Feasibility Study.* Technical Report. Carnegie-Mellon University Software Engineering Institute.

[12] Charles W. Krueger. 1992. Software Reuse. *ACM Computing Surveys (CSUR)* 24, 2 (June 1992), 131–183. https://doi.org/10.1145/130844.130856

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

[14] Luciano Marchezan, Elder Macedo Rodrigues, Maicon Bernardino, and Fábio Paulo Basso. 2019. PAxSPL: A feature retrieval process for software product line reengineering. *Software: Practice and Experience* 49, 8 (2019), 1278–1306. https://doi.org/10.1002/spe.2707

[15] Jabier Martinez, Wesley K. G. Assunção, and Tewfik Ziadi. 2017. ESPLA: A Catalog of Extractive SPL Adoption Case Studies. In *Proceedings of the 21st International Systems and Software Product Line Conference - Volume B* (Sevilla, Spain) *(SPLC '17)*. ACM, New York, NY, USA, 38–41. https://doi.org/10.1145/3109729.3109748

[16] Jabier Martinez, Tewfik Ziadi, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. 2015. Bottom-up Adoption of Software Product Lines: A Generic and Extensible Approach. In *Proceedings of the 19th International Conference on Software Product Line* (Nashville, Tennessee) *(SPLC '15)*. Association for Computing Machinery, New York, NY, USA, 101–110.

[17] Muhammad Asim Noor, Paul Grünbacher, and Christopher Hoyer. 2008. A Collaborative Method for Reuse Potential Assessment in Reengineering-Based Product Line Adoption. In *Balancing Agility and Formalism in Software Engineering*, Bertrand Meyer, Jerzy R. Nawrocki, and Bartosz Walter (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 69–83.

[18] Juliana Alves Pereira, Kattiana Constantino, and Eduardo Figueiredo. 2015. A systematic literature review of software product line management tools. In *International Conference on Software Reuse*. Springer, 73–89.

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

[20] P. Runeson and M. Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering* 14, 2 (2009), 131.

[21] Klaus Schmid. 2002. A Comprehensive Product Line Scoping Approach and Its Validation. In *Proceedings of the 24th International Conference on Software Engineering* (Orlando, Florida) *(ICSE '02)*. ACM, New York, NY, USA, 593–603.

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

[23] Karina Villela, Jörg Dörr, and Isabel John. 2010. Evaluation of a Method for Proactively Managing the Evolving Scope of a Software Product Line. In *Requirements Engineering: Foundation for Software Quality*, Roel Wieringa and Anne Persson (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 113–127.

[24] C. Wohlin, P. Runeson, M. Höst, MC. Ohlsson, and R Björn. 2012. *Experimentation in Software Engineering* (2012th ed.). Springer. 236 pages.