Investigating the Variability Impact on the Recovery of Software Product Line Architectures: An Exploratory Study

Mateus Passos Soares Cardoso
Federal University of Bahia
Salvador, Brazil
mateuspsc@dcc.ufba.br

Crescencio Lima
Federal University of Bahia
Federal Institute of Bahia
Salvador, Brazil
crescencio@gmail.com

Eduardo Santana de Almeida
Federal University of Bahia
Salvador, Brazil
esa@dcc.ufba.br

Ivan do Carmo Machado
Federal University of Bahia
Salvador, Brazil
ivanmachado@dcc.ufba.br

Christina von Flach G. Chavez
Federal University of Bahia
Salvador, Brazil
flach@ufba.br

ABSTRACT

The Product Line Architecture (PLA) of a Software Product Line (SPL) is the core architecture that represents a high-level design for all the products of an SPL, including variation points and variants. If PLA documentation is missing, it can be recovered by reverse engineering the products. The recovered PLA is a relevant asset for developers and architects, that can be used to drive specific activities of SPL development and evolution, such as, understanding its structure and its variation points, and assessing reuse. This paper presents an exploratory study that investigated the effectiveness of recovered PLAs to address variability identification and support reuse assessment. We recovered the PLA of 15 open source SPL projects using the PLAR, a tool that supports PLA recovery and assessment based on information extracted from SPL products' source code. For each project, reuse assessment was supported by existing reuse metrics. The yielded results revealed that the number of products used in PLA recovery affected the variability identification, and the number of optional features affected the components reuse rate. These findings suggest that a minimum set of representative products should be identified and selected for PLA recovery, and the component reuse rate is a candidate metric for SPL reuse assessment.

CCS CONCEPTS

• **Software and its engineering → Software product lines**; *Software architectures*; *Empirical software validation*;

KEYWORDS

Software Product Lines; Product Line Architecture; Variability; Product Line Architecture Recovery

SBCARS 2017, September 18–19, 2017, Fortaleza, CE, Brazil

ACM ISBN 978-1-4503-5325-0/17/09…$15.00
https://doi.org/10.1145/3132498.3133835

ACM Reference format:
Mateus Passos Soares Cardoso, Crescencio Lima, Eduardo Santana de Almeida, Ivan do Carmo Machado, and Christina von Flach G. Chavez. 2017. Investigating the Variability Impact on the Recovery of Software Product Line Architectures: An Exploratory Study. In *Proceedings of SBCARS 2017, Fortaleza, CE, Brazil, September 18–19, 2017,* 10 pages.
https://doi.org/10.1145/3132498.3133835

1 INTRODUCTION

Many companies – mainly in the automotive, aerospace, and electronics industries – develop a portfolio of related software products, conceived to satisfy similar, but not identical, needs of their customers. A Software Product Line (SPL) is a set of software systems that share a common and variable set of features satisfying the specific needs of a particular market segment [18]. SPL engineering supports the development and management of a product portfolio, highlighting the commonalities and variabilities, promoting reuse, and fostering customization. The adoption of the SPL paradigm brings benefits, including improved product reliability, faster time to market, and reduced costs [1].

The development of an SPL involves the implementation of different structures, processes, interfaces, and activities, therefore it is relevant for SPL engineers to pay sufficient attention to its architecture [11]. The Product Line Architecture (PLA) can be defined as (i) the core architecture that represents a high-level design for all the products of an SPL, including variation points and variants documented in the variability model [18], or (ii) an architecture for a family of products that describes the mandatory, optional, and variable components[1] in the SPL, and their interconnections [9]. The PLA is one of the most valuable SPL assets because it contains the core components of the SPL as well as the variable ones [5] in a structure that encompasses the behavior from which software products are developed [14].

Despite the benefits associated with SPL [1, 18], its development is considered expensive. For this reason, companies adopt SPL by developing a set of software products that share common characteristics, adding or removing functions from products [19]. With the growth of products portfolio, the management of variability

[1] In this paper we considered a PLA component as the concrete classes, abstract classes, and interfaces.

and reuse becomes a complex task [20], especially if there is no architecture to support it.

In this context, the PLA plays an important role to allow the SPL evolution and keep complexity under control. PLA assessment can be supported by metrics to be collected during the recovery process. For instance, reuse metrics such as those proposed by Zhang et al. [27] – Structure Similarity Coefficient (SSC), Structure Variability Coefficient (SVC) and Component Reuse Rate (CRR) [27] – and the metrics proposed by Oliveira-Junior et al. [17] – `ClassOptional`, `ClassMandatory`, and `PLTotalVariability` could be used to provide insights about the way PLA components are reused within SPL products, that may be useful for SPL developers, during maintenance tasks.

For single systems, software architecture can be recovered and documented from source code or other available information [4, 7]. Software Architecture Recovery (SAR) approaches share the goal of documenting software architecture and provide solutions to problems such as the absence of documented software architecture, and the need for detecting violations between conceptual and implemented architectures [7]. In the SPL domain the PLA provides information about the common and variable components, providing useful information for software reuse. Although the architecture description is part of the SPL adoption process, not all projects have a PLA documented. The PLA recovery can help developers with the SPL evolution and maintenance tasks.

In previous work [16], we reported the results of a literature review undertaken to investigate research work that brings together the fields of software product lines and software architecture recovery. Several approaches to PLA recovery [12, 20] were identified, as well as research trends and gaps. We have found out that few SAR tools support variability identification, an essential feature for PLA recovery. Additional features such as support for PLA assessment, were not found either. Finally, these tools were mostly not available for use. These gaps motivated us to develop the PLAR Tool, a PLA recovery and assessment tool [3].

This paper presents the results of an exploratory study conducted to assess the PLA recovered from a set of open-source SPL projects. The PLAR tool supported PLA recovery and assessment. The main contributions of this paper are (i) the recovery of the PLA from 15 SPL projects and (ii) the assessment of those 15 PLAs through reuse metrics. In this paper, we discovered that is not necessary to use all products in order to recover the PLA, the recovery process can be improved by using only the most significant configurations. The PLA recovery process described in the paper was used on SPL projects from different domains and sizes.

The remainder of the paper is organized as follows. The exploratory study conducted to investigate the recovered PLA is presented as study design (Section 2), study execution (Section 3), analysis (Section 4), and interpretation of results (Section 5). Section 6 discusses related work, and Section 7 presents concluding remarks and recommendations for future work.

2 STUDY DESIGN

In this Section, we present research questions (RQ), hypotheses, metrics, and discuss the analysis procedure defined for our study.

Table 1: GQM model for Goal 1

Goal	**Purpose**	**Verify**
	Issue	**if the number of products**
	Object	**has an impact on the PLA variability identification precision**
	Viewpoint	**from the SPL architect point of view**
Question	**RQ1**	Does the number of SPL products used in PLA recovery impact the variability identification precision in the PLA?
Metrics		SSC, SVC, CRR, Optional Features, Number of Products, and ClassOptional

We used the Goal/Question/Metric (GQM) approach [24] because measurement is defined in a top-down fashion, from goals to metrics. This study encompasses two related facets: *number of products impacting the precision to identify the variability in the PLA*, and *number of optional features impacting reuse rate*. Each facet led to different research questions and hypotheses, as discussed next.

2.1 Research Questions

We defined two GQM models in this study, each one addressing one facet. Based on the goals, we defined the research questions and related them to the set of metrics under evaluation [17, 27].

GQM Model 1: Table 1 describes the GQM model for the following goal: *"Verify if the number of products has an impact on the PLA variability identification precision"*. Related to this goal, we defined the following research question: ***Does the number of SPL products used in PLA recovery impact the variability identification precision in the PLA?***

The product generation tool provides the number of SPL products used in PLA recovery. The `PLTotalVariability` metric is used to estimate the total number of variable components expected in the PLA, and the SVC metric is used to calculate the overall variability of PLA components. Moreover, we verified the metrics values with different number of products in the comparison.

GQM Model 2: Table 2 describes the GQM model for the following goal: *"Identify correlations between the number of optional features and the component reuse rate of the PLA"*. Based on this goal we defined the following RQ: ***Is there any correlation between the number of optional features in the SPL and the component reuse rate of the recovered PLA?***

The product generation tool provides the number of optional features in the SPL. We used this information together with the SSC metric value to estimate the impact of optional features on the PLA component reuse rate.

2.2 Hypotheses

In order to answer the research questions, we postulated the following hypotheses:

RQ.1 Hypotheses

Table 2: GQM model for Goal 2

Goal	**Purpose**	**Identify**
	Issue	**if there is any correlation between**
	Object	**the number of optional features and the component reuse rate of the PLA**
	Viewpoint	**from the SPL architect point of view**
Question	**RQ2**	Is there any correlation between the number of optional features in the SPL and the component reuse rate of the recovered PLA?
Metrics		SSC, SVC, CRR, Optional Features

H_0a The number of SPL products analyzed does not influence the variability identification precision.

H_1a The variability identification precision is influenced by the number of SPL products analyzed.

RQ.2 Hypotheses

H_0b There is no relation between the number of optional features in the SPL and the component reuse rate values in the recovered PLA.

H_1b There is a relation between the number of optional features in the SPL and the component reuse rate values in the recovered PLA.

2.3 Metrics

Table 3 describes the metrics used in this study. The recovered PLA is evaluated based on the metrics analysis of its components with the SSC, SVC, and CRR metrics. The amount of common and variable components of the PLA is also measured and support the calculation of SSC, SVC and CRR.

The PLA recovery activity collects the following data from the SPL project: number of SPL optional features, number of SPL variation points, number of SPL products (M) number of common components (C_C), number of variable components (C_V), number of common relations (R_C) and number of variable relations (R_V). Data to be collected by PLA assessment includes reuse measurement values for PLA components and relations.

2.4 Analysis Procedure

The method to evaluate the variability identification precision of the recovered was based on correlation analysis. We analyzed metrics concerning both the number of products and related with the variability identification (SVC, RSVC, PLTV, ClassOptional (CO), OptionalRelation (OR), and optional-Features). The analysis examines whether the number of products (M) influences the variability identification rate of individual PLAs.

To test our hypothesis related to RQ1, we analyzed the result from the correlation analysis. As statistical tests, we applied the Spearman rank correlation [6], which is a non-parametric test that is used to measure the degree of association between two variables.

Our method for evaluating the effectiveness of the recovered PLA to support reusability evaluation was based on the analysis of CRR, which examines whether the number of optional features (OF) influences the component reuse rate of PLA.

To test our hypothesis related to RQ2, we compared the CRR values from 15 SPL projects for testing the null hypothesis that the number of optional features has no influence on the CRR. As statistical tests, we applied the ANOVA [15] to identify if at least one SPL presented different CRR value, and the Tukey test [15] to perform a pairwise comparison between the values.

3 STUDY OPERATION

3.1 Preparation

Fifteen open source SPL projects from different domains were selected for this exploratory study, based on the following criteria: lack of documented PLA and source code written in Java. Table 4 summarizes our sample and presents the number of features (mandatory and optional), classes and products of each SPL, and the tool used for product generation.

3.2 Execution

Figure 1 shows the main activities, inputs, and outputs of the PLA recovery process executed for this study. The selected SPL projects were subject to product generation, information extraction with STAN4J, and PLA recovery with the PLAR Tool.

Only valid product configurations were used. A product configuration is valid if it obeys the SPL feature model dependencies [1]. For any SPL with a potential high number of products (*e.g.* Prop4J can have 5K products), we used the T-Wise method [10] (with t = 2) to generate only a subset of SPL products. T-Wise builds only the most significant products, based on the SPL feature model. Some SPL projects (DPL, VOD, Zip Me, and GOL) had existing generated products available, so that we could skip product generation.

For each selected SPL project, we extracted a module dependency graph (MDG) based on the analysis of the products' source code, with the support of Stan4J [2]. The MDG represents the concrete classes, abstract classes, interfaces, and the relationship among them. This is done because there are different mechanisms to implement the variability (we could enlist conditional compilation, inheritance, parameterization, and overloading as the most widely used ones [2, 21]) and different composers (e.g. Featurehouse, AHEAD, CIDE, and so on [22]).

By generating the products, it was possible to recover the PLA from different SPL projects (implemented using #IFDEF directives or FeatureIDE [22]) independently of the variability implementation mechanism. The set of extracted graphs served as input to PLA recovery, with the support of the PLAR Tool [3].

The PLAR tool analyzes the MDG files to identify the variability at the architectural level by comparing the components. The main output is the PLA, represented as Module View, Class Diagram, and Design Structure Matrix (DSM). The tool also provides a metrics report, that contains the calculation of the metrics presented on table 3, which is used to assess the PLA.

[2] http://stan4j.com/

Table 3: Metrics used to evaluate the PLA.

Metric	Description	Formula	Source
SSC	SSC calculates the overall similarity between PLA components.	$\frac{\lvert C_C \rvert}{\lvert C_C \rvert + \lvert C_V \rvert}$	[27]
SVC	SVC calculates the overall variability between PLA components.	$\frac{\lvert C_V \rvert}{\lvert C_C \rvert + \lvert C_V \rvert}$	[27]
CRR	Calculates the component reuse rate of each component of the PLA	$\frac{\sum_i Ex(M_i)}{\lvert M \rvert} \times 100\%$	[27]
ClassOptional	Calculates the number of classes implementing the optional features	$\sum C_V$	[17]
ClassMandatory	Calculates the number of classes implementing the mandatory features	$\sum C_C$	[17]
PLTotalVariability	Estimates the number of variable components found on PLA	$\sum RC_V + \sum C_V$	[17]

Legend: C_C - Total number of Common Components; C_V - Total number of Variable Components; M - Total number of SPL products; RC_V - Total number of Variable Relations; $Ex(M_i)$ - returns 1, if component i is present in the product architecture, 0 otherwise.

Table 4: SPL Projects analyzed and Metrics collected for PLAs

SPL	#F	#FM	#OF	#P	#C	Gen.	SSC	SVC	RSSC	RSVC	CO	OR	CM	MR	PLTV
DPL	5	3	2	12	4	NA	0.5	0.5	0.3	0.7	2	2	2	1	4
VOD	11	6	5	32	42	NA	0.8	0.2	0.7	0.3	10	23	32	55	33
Zip Me	7	2	5	32	31	NA	0.8	0.2	0.7	0.3	6	14	25	32	20
GOL	21	12	9	65	21	NA	0.6	0.4	0.7	0.3	8	11	13	24	19
GPL	38	18	20	155	15	CD	0.6	0.4	0.4	0.6	6	23	9	16	29
Prop4J	13	0	13	31	14	FH	0.1	0.9	0.0	1.0	13	50	1	0	63
BankAccount	6	0	6	24	2	FH	1.0	0.0	1.0	0.0	0	0	2	1	0
BankAccountv2	8	0	8	72	3	FH	0.7	0.3	0.5	0.5	1	1	2	1	2
DesktopSearcher	22	6	16	462	41	AH	0.3	0.7	0.1	0.9	30	134	11	14	164
Elevator	6	0	6	20	5	FH	1.0	0.0	1.0	0.0	0	0	11	29	0
E-mail	6	0	6	40	3	FH	1.0	0.0	1.0	0.0	0	0	3	4	0
ExamDB	3	0	3	8	4	FH	1.0	0.0	1.0	0.0	0	0	4	5	0
PayCard	3	0	3	6	7	FH	0.7	0.3	0.4	0.6	2	5	5	3	7
PokerSPL	11	2	9	28	8	FH	0.5	0.5	0.3	0.7	4	5	4	2	9
UnionFind	10	2	8	6	4	FH	1.0	0.0	1.0	0.0	0	0	4	4	0

Legend: **[#F]** Features **[#FM]** Mandatory Features **[#OF]** Optional Features **[#P]** Product **[#C]** Classes **[Gen.]** Product Generator [**NA**] Not Available [**CD**] CIDE [**FH**] FeatureHouse [**AH**] AHEAD **[CO]** `ClassOptional` **[OR]** `OptionalRelation` **[CM]** `ClassMandatory` **[MR]** `MandatoryRelation` **[PLTV]** `PLTotalVariability`

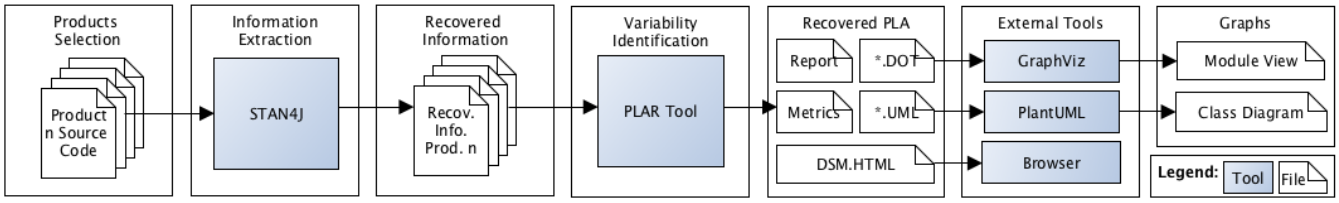

Figure 1: The overall recovery process: Activities, inputs and outputs.

3.3 Data collection

For each SPL project studied, the PLA was recovered and metrics were collected using the PLAR tool. The complete data set used in this exploratory study is available at the study website[3].

[3] https://sites.google.com/view/sbcars2017-mpassos/home

4 DATA ANALYSIS

This section presents the statistical analysis of the treatment variables relating to the data items gathered in the study. First, we present some descriptive statistics for the dependent and independent variables; next, we present the analysis of each SPL data – because the SPL projects used different techniques to implement the variability.

In order to evaluate the recovered PLA, we measured SSC, SVC, and CRR values. The SPL projects with a high SSC value and a low SVC value indicate that the PLA is mostly composed of common components. Conversely, projects with high SVC value and low SSC value indicate that the PLA is mostly composed of variable components.

4.1 Descriptive Statistics

Table 4 presents the metric results of the recovered PLAs. The SSC metric is used to calculate the overall similarity of PLA components; the maximum value is 1. The SVC metric calculates the general variability of PLA components; the maximum value is 1. The RSSC and RSVC metrics are similar to SSC and SVC, respectively. However, they are used to measure the similarity and variability of relations among PLA components.

The metrics ClassMandatory calculates the number of classes implementing the mandatory features and ClassOptional calculates the number of classes implementing the optional features. Besides, the metrics OptionalRelation and MandatoryRelation use the same principle to calculate the number of mandatory and optional relations. It is possible to perform these calculations because PLAR analyzes the classes and relations captured in the MDG files.

PLTotalVariability is a metric that estimates the PLA variability. Table 4 shows the ClassOptional and OptionalRelation metrics.

The CRR metric is missing from the overview presented in Table 4. This metric provides a measure for each PLA component and relation, as it calculates the amount of products (ratio) that have a specific component or relation. Components with CRR of 100% indicates that the component is present in all the products. Values above 50% mean that the component was used at least in half of SPL products.

The ClassVP and ComponentVariable metrics were not mentioned in Table 4. The reason is that those metrics only indicate whether a specific PLA component is either variable or not. Such information could be visualized from the output files generated by PLAR.

Figure 2 shows a boxplot with the distribution of the CRR values for each SPL. The values range from 0 to 100, in which lower values indicate the component is reused only in a small set of the products, and the higher values indicate the component is reused in a lot of products. Next, we detail the SPL projects that comprised variable elements, as observed in the recovered PLA.

4.2 Draw Product Line Results

Draw Product Line (DPL) implements five features (three mandatory and two optional) that allow the configuration of a small number of products. The SSC value indicates variability in 50% of the PLA components, while the RSVC value indicates variability in 2/3 of the PLA relations.

Table 5: CRR Measures for DPL components

Component	CRR_{pair}	CRR_8	CRR_{all}
BasicRectangle	100.0	62.5	66.7
Canvas	100.0	100.0	100.0
Line	50.0	50.0	66.7
Main	100.0	100.0	100.0

Table 5 presents the CRR values for the PLA components. The CRR_{pair}, CRR_8 and CRR_{all} columns present the CRR value for recovery based on two products, eight products, and all SPL products configurations, respectively.

The CRR measures for *Canvas* and *Main* (two classes implementing common features) are 100%. For *BasicRectangle*, the CRR_{pair} measure is 100% and variability could not be identified. The CRR_8 decreased 35.5% (reaching 62.5%) and CRR_{all} increased 4.2% (reaching 66.7%). For *Line*, the CRR_{pair} measure is 50% and some variability could be identified. The CRR_8 remains 50% and CRR_{all} increased 16.7% – reaching 66.7%.

Further investigation on these results is needed to confirm whether changes performed on DPL components that presented CRR values above 50% could impact many DPL products [12].

4.3 Video on Demand Results

Video on Demand (VOD) implements eleven features (six mandatory and five optional) that provide the creation of 32 products. The SSC value for VOD is 0.77, indicating that 32 products reused most of its PLA components. We found similar results for the PLA relations (see Table 4).

There are 32 classes implementing the mandatory features and 10 classes implementing the optional features in the VOD SPL. The classes implementing the optional features had a CRR of 50%, that is, these components were used by half of the SPL products. However, some relations (VOD boxplot outliers in Figure 2) presented a CRR of 25% and 3.25% indicating that few products used them. For this reason, the refactoring of the components involved in this relation should be considered [27]. Due to space limitation, information about the PLA and the Table with the CRR values for the 42 classes of the VOD SPL project are only available at the study website.

4.4 Zip Me Results

Zip Me has 32 generated products. The SSC measure was 0.8 indicating the components were similar in 80% of the products. The RSSC was 0.7, also a high value for reuse of relations (see Table 4).

The PLA components presented high CRR values – 25 common and 6 variable components – above 50%. The PLA relations also presented high CRR values – 32 common and 14 optional relations. From the optional relations, 11 presented a CRR value above 50%; the other relations had a CRR of 25%. These components, which presented low CRR values, are the outliers Figure 2 shows.

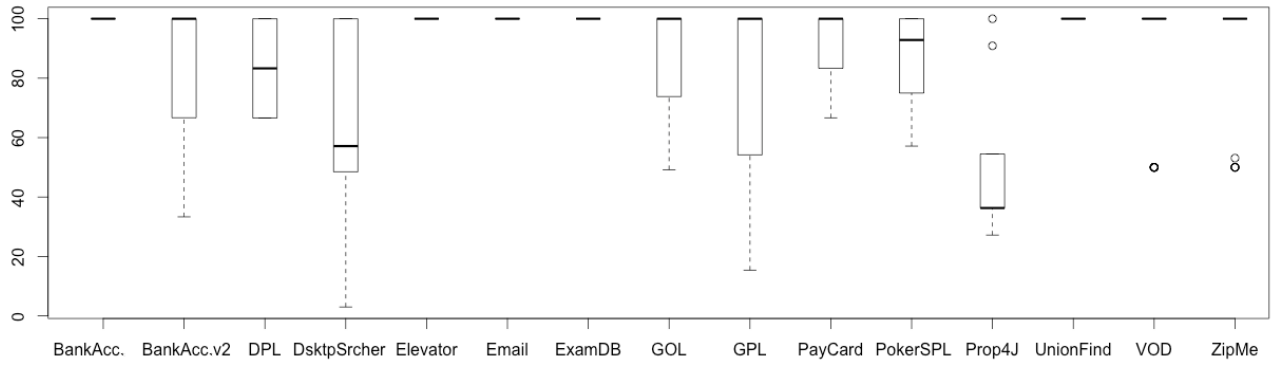

Figure 2: Boxplot of Component Reuse Rate per PLA

4.5 Game of Life Results

Game of Life (GOL) has 65 generated products. The SSC value was 0.62, indicating a larger amount of common components rather than variable ones (see Table 4).

Although the GOL results indicated low SSC values, some components were present in almost every GOL product (98% and 73%). We found similar results for CRR and PLA relations.

Figure 3 shows the DSM for GOL PLA. Rows and columns headers of a DSM are named after PLA components. The darker items represent the commonalities of the PLA while the lighter items represent the variable components. The tool colors the dependency between two common components with darker color, and between a variable component and another component using lighter color.

The GOL DSM shows that, unlike GPL and Prop4J, there is not a central node, *i.e.* a component that is related to almost every PLA component. The DSM also allows to visualize that PLA relations are scattered in the components. We also noticed that some PLA components did not have any relation; we believe this is due to features not implemented or discarded while their classes still remain in source code.

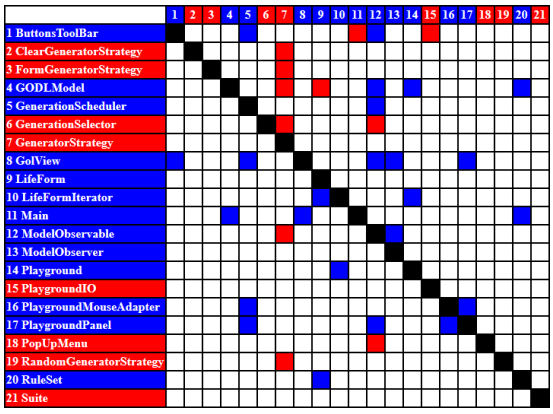

Figure 3: Design Structure Matrix for GOL

4.6 Graph Product Line Results

Graph Product Line (GPL) has 155 products. The SSC value was 0.6 indicating more common than variable components. However, the opposite happened with the relations – the SSC value was 0.42. This scenario indicated that most of the PLA relations were variable. According to Zhang et al. [27], this is a symptom of bad component reuse, suggesting a potential candidate for improvement.

The CRR values were high. However, we identified that some components, such as *CycleWorkSpace* and *GlobalVarsWrapper*, presented a low CRR value. Furthermore, the majority of relations presented low CRR values. The components *Graph* and *Vertex* have relationships with all the other components. It deserves further investigation whether these two classes present the God class smell [8].

4.7 Prop4J Results

Prop4J has no mandatory features, implementing only optional features. In addition, for the absence of mandatory features, this SPL project allowed the generation of 5029 possible configuration of products based on optional features. For this reason, we instantiated 11 products using T-Wise method configuration [10]. After the PLA recovery based on a subset of 11 products, we identified only one common component (*Node*) and 13 variable components. All the recovered relations are variable. Figure 2 also reflected this information. For instance, *Node* is one outlier of the Prop4J boxplot.

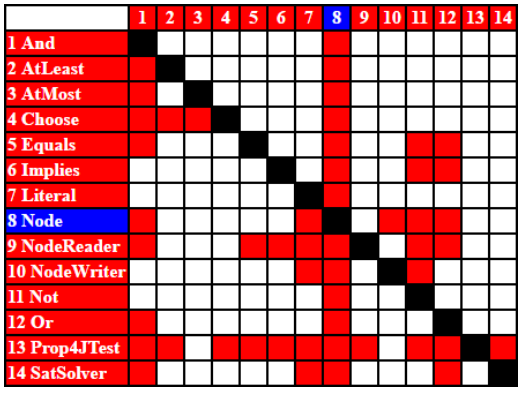

Figure 4: Design Structure Matrix for Prop4J

Figure 4 shows the DSM for Prop4J PLA. The *Node* component is common to the 11 products analyzed. However, the relations were variable confirming the CRR values. These results suggest that the PLA needs maintenance to improve the CRR values.

In Prop4J DSM, we identified two central classes, *Node* and *Prop4J test* that relate to almost every class present in the PLA, which is an indicative of the God class smell [8] and require further investigation.

4.8 BankAccountV2 Results

BankAccountV2, the second version of the BankAccount SPL, introduced new features to the SPL. The implementation of these new features required new classes and relations, which required the identification of variability in classes and relations that were absent from the previous version of this SPL. The SSC value was 0.7, meaning that most of components and relations are common to 70% of the products.

4.9 DesktopSearcher Results

DesktopSearcher allowed the generation of 462 products. The values for SSC and RSSC were considered low [27]. The SSC value was 0.26 which means that almost all of its classes are variable among the products. The CRR presented the same behavior. In most cases, the values were lower than 0.5.

Theses values indicated that the SPL products tend to have exclusive products that require specific features in only part of the products [1] which we believe to be the cause of the low CRR values found in this SPL project.

4.10 PayCard Results

PayCard is a small-sized SPL project, with respect to the amount of product configurations, and classes. The project presents a value of 0.71 for SSC, *i.e.*, most of its classes are common to all products. However, the dependencies among classes presented a value of 0.37 for RSSC, which means that most of the relations are variable.

4.11 PokerSPL Results

PokerSPL is another small-sized SPL project, regarding the number of classes and products. It presented a 0.5 SSC value meaning that half of its classes are common to all products. The CRR values could be higher since most of the variability in this SPL was found on the values assumed by some of its classes attributes.

Accordingly, the CRR values for the SPL relations present similar results. A possible explanation is that most of the SPL variability is implemented in the values of class attributes.

5 DISCUSSION

In this section, we interpret the results and discuss the findings by answering the research questions.

5.1 Answers to the Research Questions

For the first research question, we verified if *the number of products has an impact on the PLA variability identification precision.* Figure 5 shows SSC (darker color) and SVC (lighter color) metrics of four SPL projects (DPL, VOD, Zip Me, and Prop4J) collected during different stages of comparison. We selected these projects randomly from the sample. All the SPL projects presented the same patterns.

We identified that the precision regarding the variability identification increased when we included more products in the comparison. This happens because, with more products, more configurations are analyzed. By analyzing all the feature combinations, it is possible to guarantee the detection of all the variable components and provide a reliable PLA.

Moreover, after a certain number of comparisons, the value of the metrics became constant. For instance, we compared 18 products aiming to recover all the variability details of the Zip Me SPL. We observed the same pattern on other SPL projects. For example, it was necessary to compare 17 products to recover all the variability details of the VOD SPL.

As the PLA recovery process examined and merged more products, the set of components and relations that comprise the PLA and metrics values tends to stabilize. The set of products (after the metrics stabilization) had a common structure, with variations present in low-level details. We also identified this pattern when we analyzed the CRR values with different combination of products in the recovery.

Table 6 presents the CRR values from the Prop4J project. The CRR_{pair}, CRR_8 and CRR_{all} columns present the CRR value for recovery based on two products, eight products, and all SPL products, respectively. As we raised the number of products in the comparison, the CRR precision became higher.

Table 6: CRR Measures for Prop4J

Component	CRR_{pair}	CRR_8	CRR_{all}
And	100.0	50.0	54.5
AtLeast	50.0	37.5	36.4
AtMost	50.0	37.5	36.4
Choose	50.0	37.5	36.4
Equals	100.0	37.5	27.3
Implies	100.0	50.0	54.5
Literal	100.0	87.5	90.9
Node	100.0	100.0	100.0
NodeReader	100.0	62.5	54.5
NodeWriter	50.0	75.0	54.5
Not	50.0	50.0	36.4
Or	50.0	37.5	36.4
Prop4JTest	50.0	37.5	36.4
SatSolver	50.0	37.5	45.4

Moreover, to answer the RQ1, in the first stage of the analysis, we performed a correlation analysis among the variables of the exploratory study (see Figure 6). We identified a positive correlation between the number of products and the metrics addressing the variability (CO, OR, PLTV, SVC, and number of optional features) which means that with the increase of the number of optional features we have a increase in the variability identified on the PLA.

We also identified a negative correlation between the number of optional features and the metrics related to commonality (SSC and RSSC) which means that with the increase of optional features

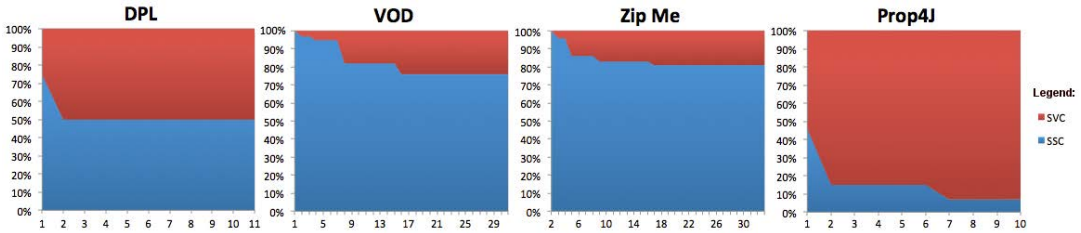

Figure 5: SSC and SVC metrics according to the number of products in the comparison

that commonality tends to decrease. Table 7 shows the *Spearman* correlation test that rejected the null hypothesis.

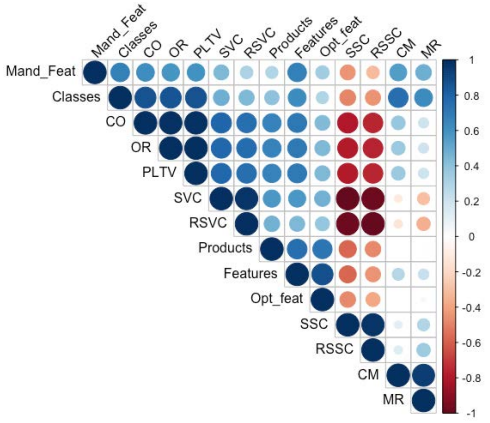

Figure 6: Correlation analysis

Table 7: Comparisons that rejected the null hypothesis RQ1

Comparison	p-value
Products-CO	7.0e-03
Products-OR	7.0e-03
Products-PLTV	7.0e-03
Products-SVC	2.0e-02
Products-OF	2.0e-03

In the second research question, we investigated if *there is a relation between the number of optional features in the SPL and the component reuse rate of the recovered PLA*, we used the ANOVA (Analysis of Variance) to test the variables and the p-value was 1.3e-07. Such evidence allows to reject the null hypothesis (H_{0a}) of equal population means. Therefore, it is possible to conclude that at least one PLA has CRR values significantly different from the others.

To identify the different means, we applied the Tukey test. We performed and analyzed 105 comparisons, and only 12 of them presented statistically significant differences. Table 8 shows p-values of the comparisons that rejected the null hypothesis (the SPLs involved in the test, and the p-value). Based on such data, we identified that Prop4J (in eight comparisons) and DesktopSearcher (in four comparisons) yielded statistical difference in CRR values among the PLAs (see Figure 2).

Table 8: Comparisons that rejected the null hypothesis RQ2

ID	Comparison	p-value
40	Elevator-DesktopSearcher	2.3e-03
43	GOL-DesktopSearcher	4.4e-02
49	VOD-DesktopSearcher	2.1e-03
50	ZipMe-DesktopSearcher	1.5e-03
57	Prop4J-Elevator	4.0e-05
74	Prop4J-ExamDB	1.9e-02
81	Propo4J-GOL	8.0e-04
92	Prop4J-PayCard	2.2e-02
96	Prop4J-PokerSPL	4.4e-02
100	UnionFind-Prop4J	1.9-e02
101	VOD-Prop4J	7.0e-05
102	ZipMe-Prop4J	4.0e-05

According to the correlation analysis performed (see Figure 6), the use of optional features impact the CRR values of the PLA by decreasing the commonality and increasing the variability. Configuring a product with optional features implies in the appearance of new classes associated with these features. Meaning that there will be new variable components in the product architecture, that are often associated with specific product configurations.

5.2 General Findings

Correlation between metrics. From the overall results for the fifteen SPL projects, we noticed that when the value of SSC was high, the PLA components presented high CRR values as well. This may be a preliminary evidence for a correlation between SSC and CRR metrics.

Some metrics provided support for other metrics. The quantitative metrics `ClassMandatory` and `ClassOptional` counted the number of classes implementing mandatory features and optional features. They confirmed the SSC and SVC metrics values. Moreover, `ClassVP` and `ComponentVariable` indicated if a given class or component presented a variability.`ClassMandatory`, `CLassOptional`, `ClassVP` and `ComponentVariable` provided support for the SSC, SVC, and CRR metrics. The former validated and confirmed the values of the latter.

Feature scattering and component reuse rate. In projects with metrics high values (Zip Me, VOD, and GOL respectively), the classes

outnumbered the features, with feature scattering in a significant amount of classes. To perform this analysis, we verified how the feature selection to build each product was spread through the source code manually and compared to the metrics collected by the PLAR. The relation between feature scattering and high CRR deserves further investigation.

5.3 Threats to Validity

The following threats to validity are discussed to reveal their potential interference with our study design.

Internal Validity. PLAR tool limitations [3] may have impacted the results of the exploratory study. For instance, the input for PLAR tool is a MDG file created by STAN4J and Analizo. Currently, the extracted MDG only supports "call" dependencies between modules. Inheritance relationships are not extracted.

External Validity. No industrial SPL projects were used in this exploratory study; only open source SPL projects created for educational and research purposes were used. To minimize such a threat, we analyzed projects widely-accepted by SPL research community, which have been used as testbeds for empirical evaluations [12].

Construct Validity. The recovered PLA from the SPL projects were not verified by SPL developers. To minimize this threat, we performed a manual PLA extraction, which served as an oracle. Then, we compared the number of variable and common elements (including components and its relationships) obtained by PLAR tool against the oracle. The results indicated that there was no difference in the number of elements detected through the tool and manually.

Conclusion Validity. We used statistical algorithms as recommended by Wohlin et al. [25] on experimentation in software engineering to computing the statistical significance and strength of the relationship between the metrics. These procedures aim to minimize issues regarding the conclusions we draw. Two authors checked the analysis to avoid missing data and prevent biases.

The metrics SSC and CRR present an overview of the common and variable components of a PLA by showing the presence of the components among all the products. However, low-level granularity variability in the components, such as variable with different values and different methods implementation, are not covered by these metrics. This information can give a different perspective about the CRR.

6 RELATED WORK

Wu et al. [26] presented a semi-automatic PLA recovery approach. The authors defined measures to detect similarity and variability points on software products source code of the same domain to migrate to the SPL paradigm. The study reports on a case study carried out with an industrial product line. The assumption is that legacy products of a same domain have similar designs and implementation that can be used to build a SPL. In our approach, we build the PLA from the products generated by the SPL project.

Losavio et al. [13] proposed a reactive refactoring bottom-up process to build a PLA from existing similar software product architectures of a domain. The main assets were expressed by UML logical views. Their work is focused on the construction and representation of a candidate PLA followed by an optimization process to obtain the final PLA. The refactoring process was applied to a case study in the robotics industry domain. The focus of our work is the assessment of recovered PLA based on metrics analysis.

Torkamani [23] presents a novel SPL quality attribute for called Extractability. The attribute is calculated based on the weight of the reusable component over the weight of all components, a process very similar to the CRR metric calculation. Extractability effectiveness on six SPLs from a iranian telecommunication company was evaluated in practice. In our study, we analyzed SPLs from different domains.

7 CONCLUDING REMARKS

Product Line Architecture recovery provides useful information for SPL developers and architects, to support maintenance, understand the implementation of SPL variability and foster reuse. The recovered PLA components and their relationships can serve as a basis for different types of visualization that expose variability, and for different types of analysis (*e.g.* identification of components that are more likely to be reused, propagation analysis during the implementation of reuse changes).

In this paper, we presented the results of an exploratory study to assess the recovered PLAs from 15 open source SPL projects implemented in Java. The PLAR tool was used to support PLA recovery with the identification of commonality and variation points.

Eleven out of fifteen recovered PLAs had high CRR values indicating high reuse of components during the SPL development phase. The results provided initial evidence regarding a correlation between the metrics values and the components reuse rate. We aim to replicate this study by including more SPL projects to strengthen the evidence base.

One contribution of this paper was the PLA recovered for each SPL project, because none of them presented Product Line Architecture documentation. The results of this exploratory study can be used to improve the design and execution of future empirical studies.

As future work, we plan to evolve the PLAR tool to address existing problems and limitations. For instance, our main focus is to optimize the recovered PLA to address other types of variability not mapped, such as variability on the class attribute and method level. Also, we are working on improving the PLA visualization by mapping the SPL features on the PLA.

Furthermore, the question about the minimum subset of products that covers PLA recovery and results in an architecture that documents the SPL shall be investigated.

ACKNOWLEDGMENTS.

The authors would like to thank the anonymous reviewers for the thorough feedback. This work is partially supported by FAPESB grants BOL1564/2015, BOL2443/2016 and JCB0060/2016, and INES, grant CNPq/465614/2014-0.

REFERENCES

[1] Sven Apel, Don Batory, Christian Kstner, and Gunter Saake. 2013. *Feature-Oriented Software Product Lines: Concepts and Implementation.* Springer Publishing Company, Incorporated.
[2] Jan Bosch and Rafael Capilla. 2013. *Variability Implementation.* Springer Berlin Heidelberg, 75–86.
[3] Mateus Passos Soares Cardoso, Crescencio Lima, Christina von Flach Garcia Chavez, and Ivan do Carmo Machado. 2017. PLAR Tool - A Sofware Product

Line Architecture Recovery Tool. In *8th Brazilian Conference on Software: Theory and Practice - Tools Session*. 18–22.

[4] Paul Clements, David Garlan, Len Bass, Judith Stafford, Robert Nord, James Ivers, and Reed Little. 2002. *Documenting Software Architectures: Views and Beyond*. Pearson Education.

[5] Thelma Elita Colanzi and Silvia Regina Vergilio. 2013. Representation of Software Product Line Architectures for search-based design. In *2013 1st International Workshop on Combining Modelling and Search-Based Software Engineering (CMSBSE)*. 28–33.

[6] W.W. Daniel. 1990. *Applied nonparametric statistics*. PWS-Kent Publ.

[7] Stephane Ducasse and Damien Pollet. 2009. Software Architecture Reconstruction: A Process-Oriented Taxonomy. *IEEE Transactions on Software Engineering* 35, 4 (July 2009), 573–591.

[8] Martin Fowler. 1999. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, Boston, MA, USA.

[9] Hassan Gomaa. 2004. *Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

[10] Christopher Henard, Mike Papadakis, Gilles Perrouin, Jacques Klein, Patrick Heymans, and Yves Le Traon. 2014. Bypassing the Combinatorial Explosion: Using Similarity to Generate and Prioritize T-Wise Test Configurations for Software Product Lines. *IEEE Transactions on Software Engineering* 40, 7 (July 2014), 650–670.

[11] Frank J. van der Linden, Klaus Schmid, and Eelco Rommes. 2007. *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Springer-Verlag.

[12] Lukas Linsbauer, Roberto Erick Lopez-Herrejon, and Alexander Egyed. 2016. Variability extraction and modeling for product variants. *Software & Systems Modeling* (Jan 2016).

[13] Francisca Losavio, Oscar Ordaz, Nicole Levy, and Anthony Baiotto. 2013. Graph modelling of a refactoring process for Product Line Architecture design. In *XXXIX Latin American Computing Conference (CLEI)*. 1–12.

[14] Elisa Yumi Nakagawa, Pablo Oliveira Antonino, and Martin Becker. 2011. *Reference Architecture and Product Line Architecture: A Subtle but Critical Difference*. Springer-Verlag, Essen, Germany, 207–211.

[15] Mary Natrella. 2010. *NIST/SEMATECH e-Handbook of Statistical Methods*. NIST/SEMATECH.

[16] Crescencio Rodrigues Lima Neto, Mateus Passos Soares Cardoso, Christina von Flach Garcia Chavez, and Eduardo Santana de Almeida. 2015. Initial Evidence for Understanding the Relationship between Product Line Architecture and Software Architecture Recovery. In *IX Brazilian Symposium on Components, Architectures and Reuse Software*. 40–49.

[17] Edson Alves Oliveira-Junior, I Gimenes, and J Maldonado. 2008. A metric suite to support software product line architecture evaluation. In *XXXIV Conferencia Latinoamericana de Informatica*. 489–498.

[18] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. 2005. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag.

[19] Julia Rubin and Marsha Chechik. 2012. Locating distinguishing features using diff sets. In *2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. 242–245.

[20] Anas Shatnawi, Abdelhak Seriai, and Houari Sahraoui. 2014. Recovering Architectural Variability of a Family of Product Variants. In *Software Reuse for Dynamic Systems in the Cloud and Beyond: 14th International Conference on Software Reuse*. Springer, 17–33.

[21] Mikael Svahnberg, Jilles van Gurp, and Jan Bosch. 2005. A Taxonomy of Variability Realization Techniques: Research Articles. *Softw. Pract. Exper.* 35, 8 (2005), 705–754.

[22] Thomas Thüm, Christian Kästner, Fabian Benduhn, Jens Meinicke, Gunter Saake, and Thomas Leich. 2014. FeatureIDE: An extensible framework for feature-oriented software development. *Science of Computer Programming* 79 (2014), 70–85.

[23] Mohammad Ali Torkamani. 2014. Extractability Effectiveness on Software Product Line. *International Journal of Electrical and Computer Engineering* 4, 1 (2014), 127.

[24] Rini van Solingen, Vic Basili, Gianluigi Caldiera, and H. Dieter Rombach. 2002. Goal Question Metric (GQM) Approach. In *Encyclopedia of Software Engineering*. John Wiley & Sons, Inc.

[25] Claes Wohlin, Per Runeson, Martin Hst, Magnus C. Ohlsson, Bjrn Regnell, and Anders Wessln. 2012. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated.

[26] Yijian Wu, Yiming Yang, Xin Peng, Cheng Qiu, and Wenyun Zhao. 2011. Recovering Object-oriented Framework for Software Product Line Reengineering. In *Proceedings of the 12th International Conference on Top Productivity Through Software Reuse (ICSR'11)*. Springer-Verlag, Berlin, Heidelberg, 119–134.

[27] Tao Zhang, Lei Deng, Jian Wu, Qiaoming Zhou, and Chunyan Ma. 2008. Some metrics for accessing quality of product line architecture. In *International Conference on Computer Science and Software Engineering*, Vol. 2. IEEE, 500–503.