Empirical Software Engineering (2022) 27:166
https://doi.org/10.1007/s10664-022-10198-3

Intensifying the search-based optimization of product line architectures with crossover operators

Diego Fernandes da Silva[1] · Luiz Fernando Okada[1] · Wesley K. G. Assunção[2,3] · Thelma Elita Colanzi[1]

Accepted: 2 July 2022 / Published online: 20 September 2022

Abstract

The Product Line Architecture (PLA) is a crucial artifact for the development of Software Product Lines. However, PLA is a complex artifact to be designed due to its large size and the multiple conflicting properties that need to be considered to ensure its quality, requiring a great effort for the architect. PLA designing has been formulated as an optimization problem aiming at improving some architectural properties in order to maximize both the feature modularization and the relational cohesion, and to minimize the class coupling. This kind of problem was successfully solved by multi-objective evolutionary algorithm. Nevertheless, most of existing approaches optimize PLA designs without applying the crossover operator, one of the fundamental genetic operators. To overcome these limitations, this paper aims to intensify the search-based PLA design optimization by presenting three crossover operators. These operators were empirically evaluated in quantitative and qualitative studies using three well-studied PLA designs. The experiments were conducted with eight experimental configurations of NSGA-II in comparison with a baseline that uses only mutation operators. Empirical results showed that there are significant differences among the use of only mutation and mutation with crossover. Also, we observed that the crossover operators contributed to generate solutions with better feature modularization. Finally, we could see that the proposed operators complement each other, since the experiment that combines at least two of the proposed operators achieved better results.

Keywords Multi-objective evolutionary algorithm · Software product line · Software architecture · Recombination operators

Communicated by: Aldeida Aleti, Annibale Panichella, Shin Yoo

This article belongs to the Topical Collection: *Advances in Search-Based Software Engineering (SSBSE) 2020*

✉ Diego Fernandes da Silva
dizzusmon@gmail.com

✉ Thelma Elita Colanzi
thelma@din.uem.br

[1] State University of Maringá (UEM), Maringá, Brazil

[2] DI – Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro, Brazil

[3] PPGComp – Western Paraná State University (Unioeste), Cascavel, Brazil

1 Introduction

The architecture of a software system comprehends its components and the relationship between them (Bass et al. 2003). Defining a proper architecture is paramount to develop a system (Gomaa 2011). For developing a family of software products, Product Line Architecture (PLA) is one of the main artifacts to be defined for the Software Product Line (SPL). A PLA describes all available features for the family and presents the architectural variations for creating SPL products (Van der Linden et al. 2007). However, designing a PLA is a non-trivial activity, since SPL architects should consider important, sometimes conflicting, properties such as extensibility, reusability and modularity (Colanzi et al. 2014). These properties play an important role in the PLA design to maximize the reuse, maintenance, and evolution of an SPL. For example, feature modularization must be taken into account during the design of a PLA, as it directly impacts on SPL extensibility and reusability (Nunes et al. 2009). Modularity also affects cohesion, coupling, crosscutting, and tangling among components in a PLA (Czarnecki 2013). The higher feature modularization, the lower scattering functionality in PLA design. Scattering functionalities negatively impact PLA extensibility. A proper PLA is important for the whole SPL life cycle, since it supports the configuration of many products (Van der Linden et al. 2007).

To aid the designing of PLAs, e.g., dealing with variability and modularization, optimization approaches with multi-objective evolutionary algorithms (MOEAs) have been successfully applied (Colanzi et al. 2014; Colanzi and Vergilio 2016; Choma Neto et al. 2019). MOEAs are bio-inspired computational intelligence metaheuristics based on the process of natural selection (Deb 2015). Traditionally, MOEAs apply crossover and mutation operators to perform changes in an individual, i.e., solution, with the goal of performing an evolutionary process of optimization. These changes evolve an existing PLA design to improve it with regard to given properties, such as feature modularization, relational cohesion and class coupling (Colanzi et al. 2014). However, the use of crossover operator has been poorly investigated for designing PLAs (discussed in related work, Section 2). The crossover operator is arguably one of the fundamental operators of evolutionary algorithms (Goldberg 1989). This operator combines parts of two different individuals (parents) to create offspring. In the context of architectures for traditional single software systems, crossover operators have been used to merge two alternative designs in order to take the best parts of them (Räihä et al. 2009). Yet, empirical results of a study about software design revealed that exclusively using the mutation operator lead to a very homogeneous population of solutions, quickly reaching a local optimum (Räihä et al. 2009). This confirms the importance of the crossover operator to combine genes for maintaining diversity within the population. Hence, the crossover operator is crucial to the evolutionary process for promoting diversity. This is the main motivation of our work.

The majority of existing studies do not apply crossover operator to optimize PLA designs (Colanzi et al. 2014; Choma Neto et al. 2019; Choma Neto et al. 2018; Guizzo et al. 2017). Only one study introduced a crossover operator, called *Feature-driven Crossover*, which has the goal of maintaining and improving feature modularization in PLA designs (Colanzi and Vergilio 2016) (further details in Section 3). A qualitative analysis of the application of this crossover operator concluded that fitness and diversity of solutions are improved. However, a side effect was observed, which is the generation of some incomplete solutions, with methods and attributes missing in classes of the resulting PLAs (Colanzi and Vergilio 2016). To fill this gap in the research and practice of optimizing PLAs, in a previous work we presented two effective crossover operators to enhance the search-based PLA design optimization (Silva et al. 2020). Specifically, we proposed a new version of *Feature-driven*

Crossover (Colanzi and Vergilio 2016) correcting the problem of incomplete solutions, and adapt the *Complementary Crossover* (Räihä et al. 2010) that focus on creating a solution so that parents complement each other, to the PLA design context (Silva et al. 2020). In an in-depth quantitative and qualitative evaluation, we observed that the benefit of using crossover operators is to combine/merge parts already optimized in different individuals. These operators lead to more diversity in the population of potential PLA designs. In addition, crossover operators generated solutions with better feature modularization (Silva et al. 2020).

In this present paper, we present an extension of our previous work (Silva et al. 2020). The goal of our work reported in this extension is to intensify the search-based PLA design optimization by providing three effective crossover operators (described in Section 4). In this extension, we present a third new crossover operator for the optimization of PLAs, namely *Modular Crossover*. Also, we present an empirical study (Section 5) with three subject PLAs to quantitatively evaluate the performance of the PLA optimization when using crossover operators, providing an additional real-world subject system to enrich analysis and the generalization of the results. We evaluate experimental configurations when using each crossover operator individually, as well as all the combination of crossover operators, summing eight experimental configurations, five more than in our previous work. For a qualitative evaluation, we investigate the impact of the crossover operators regarding the feature modularization of the PLA design solutions. Furthermore, as part of this extension, we conducted a qualitative study with five software engineers, experts in SPL Engineering and two of the subject PLAs, to collect their opinion about characteristics of obtained solutions. Finally, we performed an extended analysis and discussion of the results.

Summarizing, the main contributions of this paper are: (i) the improved version of *Feature-driven Crossover* to avoid incomplete or inconsistent solutions; (ii) two additional crossover operators for PLA design, namely *Complementary Crossover* and *Modular Crossover*; (iii) an in-depth quantitative and qualitative evaluation of the proposed crossover operators involving three SPLs, which shows the positive impact of the proposed operators to PLA design (Section 6); (iv) a qualitative analysis of solutions generated for two SPLs by experts in SPL engineering, where the solutions obtained using combinations of the crossover operators were the most accepted ones; and, (v) the empirical results clearly show that the proposed crossover operators complement each other, achieve more diversity of solutions and enable the generation of solutions with better feature modularization.

2 Related Work

For covering the existing literature on the topic of this work, we performed a systematic review of studies, according to the guidelines presented by Kitchenham and Charters (2007). The main goal was to identify existing crossover operators for software design and how they impact the generation of solutions. The protocol of the systematic review as well as the obtained results were made available in the supplementary material.[1]

For conducting the systematic review study, based on the aforementioned goal, we designed a search string composed of three set of keywords:[2] (i) software architecture, (ii) evolutionary

[1]The supplementary material is available at: https://doi.org/10.5281/zenodo.6516279

[2]Example of search string with synonyms and complementary keywords: *("Software Design" OR "Software Architecture") AND ("Evolutionary Algorithm" OR "Genetic Algorithm") AND ("Crossover" OR "Crossover Operator" OR "Recombination")*

algorithms, and (iii) crossover operator. The search string was used for retrieving primary sources from six digital libraries.[3] In addition to digital libraries, we also conducted manual searches on Google Scholar[4] and backwards snowballing (Wohlin 2014). All the searches retrieved 906 results, from which we selected five papers aligned with our review goal. The selection of these works was carried out through the application of exclusion and inclusion criteria, such as the type of representation of the problem and the possibility of adaptability to the PLA design context. Other inclusion criteria were: peer-reviewed papers, text in English, focus on optimization of software design, and describing crossover operators.

The majority of studies were excluded because they do not focus on software design or PLA design (>70%). Around 13% were excluded because, in spite of dealing with software design optimization, they did not present a crossover operator proposal or implementation. Around 7% of the collected studies present a crossover operator but they are not adaptable for the context of the software design. Thus, the exclusion in large numbers is due to these restrictions and the difficulty of finding works that use a crossover operator that can be adapted to the context of optimization of PLAs, taking into account the PLA encoding adopted by MOA4PLA.

Regarding the five papers accepted in our SLR, four of them present crossover operators for software design optimization or PLA design optimization. Only one study was included because it uses a representation suitable to be adapted to a UML class diagram. The crossover operators identified in our review, their characteristics and impact on the generated solutions, are described next.

For the optimization of traditional software architectures, i.e., single systems, Simons et al. (2010) presented a strategy based on an interactive evolutionary algorithm to optimize object-oriented class design. This evolutionary algorithm applies a *Trans-Positional Crossover* operator that randomly chooses and swaps attributes and methods between two parents, based on their class position. The only constraint is that each class in the design must contain at least one attribute and one method. Thus, this positional swapping can only occur for a class that would not be empty of attributes or methods after the application of the crossover.

Instead of randomly selecting attributes and methods, Räihä et al. (2010) proposed a more elaborated crossover to preserve quality attributes present in different individuals, i.e., software architectures. These authors presented the *Complementary Crossover* operator that chooses and combines two parents having an optimized structure for given quality attributes. The rationale is to preserve and combine good quality attributes from the parents in the children. For example, an architecture with good modifiability and another architecture with good reusability are selected as parents so that the children might inherit both desirable qualities. The *Complementary Crossover* was implemented in two versions, namely *Simple Complementary Crossover* and *Complementary Gene-Selective Crossover*. The first one applies a random crossover point to the parents, whereas the second version finds the best crossover point in order to use the optimal portion of both parents. Experimental results based on fitness of the solutions showed that the two versions of the *Complementary Crossover* lead to more flexible and elaborated architectures. However, although a qualitative evaluation was presented, there was no in-depth analysis involving experts.

[3]Digital libraries commonly used for Software Engineering literature: IEEE Xplore, ACM Digital Library, Scopus, Springer Link, Engineering Village, and ScienceDirect.

[4]Google Scholar indexes scholarly literature: https://scholar.google.com/

Despite generating better architectures in terms of fitness functions, the operators presented by Simons et al. (2010) and Räihä et al. (2010) do not guarantee a proper modularization of architectural elements. Since their crossover operators does not focus on modularization, some resulting architectures present scattering of elements assigned to implementation of features.

To deal with this problem, Harman and Hierons (2002) introduced a *Modular Crossover* operator, which is based on a clustering algorithm. This operator focuses on preserving allocations of complete or partial modules from a parent to a child. The Modular Crossover Operator operates by randomly choosing similar modules in the two parents, instead of selecting an arbitrary crossover point. Then, the components of the chosen modules are exchanged between both parents. This mechanism implies the idea of integral preservation of at least one parent module in the children, and that portions of other modules are also retained.

None of the aforementioned operators were designed to consider architectures of SPLs, dealing with feature modularity. To tackle this limitation, inspired by the Modular Crossover operator presented by Harman and Hierons (2002), Colanzi and Vergilio (2016) proposed the *Feature-driven Crossover* operator. This operator is applied by randomly selecting a feature of the SPL, and then creating children by swapping the architectural elements (classes, interfaces, operations, etc.) that realize the selected feature. The rationale is to create children that combine architectural elements that better modularize SPL features that were inherited from their parents. In an empirical study to evaluate the benefits of the *Feature-driven Crossover* to PLA design optimization (Colanzi and Vergilio 2016), the authors concluded that resulting architectures in fact had a good feature modularization. However, despite the noticeable improvement in the exploration of the search space with this operator, the authors discussed that inconsistent solutions were generated. These inconsistent solutions lead to a degradation of the evolutionary process, as the individuals of these solutions were removed from the population. In this sense, the authors acknowledged the need of improvements in the implementation of the operator to reach better solutions along the evolutionary process.

In summary, by the results of our systematic review study, we could observe that architectural optimization is an active topic of research. Several studies deal with optimization of traditional software architectures (Simons et al. 2010; Räihä et al. 2010; Harman and Hierons 2002; Colanzi and Vergilio 2016) and only a few of them focus on PLAs (Colanzi and Vergilio 2016; Féderle et al. 2015). Based on the analysis of these pieces of work, we observed that only one crossover operator for PLAs were proposed (Colanzi and Vergilio 2016), which occasionally generates inconsistent solutions. We can clearly see that there is a need for more crossover operators to improve the optimization of PLA designs, leading to better development of SPLs.

3 Search-based Design of PLA

One of the most prominent approaches to optimize PLA designs is the Multi-Objective Approach for Product-Line Architecture Design (MOA4PLA) (Colanzi et al. 2014). The several objective functions, which composes an evaluation model (Verdecia et al. 2017), are based on software metrics that provide information about architectural properties relevant

to a PLA design. The objective functions currently supported by MOA4PLA are: feature modularization, SPL extensibility, variability, coupling, and cohesion.

3.1 Representation

The input for MOA4PLA is a PLA designed as a UML class diagram. This diagram must have each architectural element associated to the feature(s) that it realizes. The associations between elements and features are described by using UML stereotypes. For allowing the optimization with evolutionary algorithms, the PLA is represented using a metamodel (Fig. 1) that contains all architectural elements such as components, classes, attributes, methods, interfaces, operations, interrelationships, variabilities, variation points, and variants (Colanzi et al. 2014).

In this context, an individual is composed of the architectural representation of a PLA design represented by a class diagram relying on the metamodel of Fig. 1. The initial population is formed by *n* individuals, in which each individual is obtained by applying a mutation operator selected at random on the original PLA design. Over generations, new solutions are created by the application of crossover and mutation operators. These solutions are added to the population and the best ones are selected to be kept in the population.

The output of the optimization process is a set of design solutions, which are decoded as class diagrams, that consists of alternative designs for the PLA given as input. This set contains solutions with the optimized trade-off among the objectives that are optimized. Based on this set of alternative solutions, the software architect can decide which to adopt for the SPL development.

3.2 Genetic Operators

The PLA optimization is performed with evolutionary algorithms by successively applying genetic operators in an evolutionary process. These genetic operators are mutation and

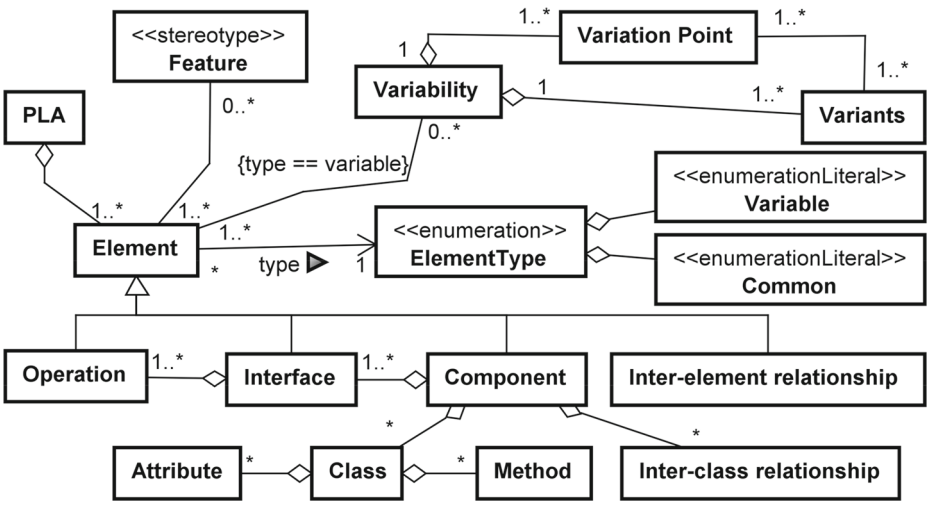

Fig. 1 PLA Metamodel (Colanzi et al. 2014)

crossover operations responsible for moving architectural elements (methods, attributes, operations), adding elements (class, package) or modularizing features in a way that positively impacts some property of the solution.

The piece of work that introduced MOA4PLA presented six mutation operators specific for PLA design (Colanzi et al. 2014), namely *Move Method*, *Move Attribute*, *Add Class*, *Move Operation*, *Add Package Manager*, and *Feature-driven Mutation*. The *Feature-driven Mutation* operator aims to improve feature modularization of a PLA. This mutation operator helps to obtain PLA designs with less scattered and tangled features, improving the feature cohesion of the architectural packages. The movements are performed taking into account the architectural layer of the elements. In addition to the mutation operators, an extension of the MOA4PLA presented a crossover operator named *Feature-driven Crossover* (Colanzi and Vergilio 2016), which is explained in details in Section 4.2.1.

3.3 Fitness Functions

The evaluation model used by MOA4PLA provides a set of objective functions. The approach allows users to select the functions related to the properties that they want to optimize. The complete descriptions and equations to compute each objective function of the evaluation model are presented in the work of Verdecia et al. (2017). In the context of our work, the three objective functions used are: COE (Relational Cohesion), ACLASS (Class Coupling), and FM (Feature Modularization).

COE measures the cohesion of a PLA design in terms of internal relationship of the design classes, measured by the H metric[5] (1). Basically, the H metric measures the number of internal relationships per class in a component. ACLASS evaluates the coupling of a class by computing the number of architectural elements that depend on other design classes ($CDepIn$), added to the elements number in which each class depends on ($CDepOut$), as presented in (2). In the case of (1) and (2), c is the number of classes.

FM evaluates the feature modularization of a solution based on the sum of the metrics for feature scattering ($CDAC$, $CDAO$, $CDAI$), feature interlacing ($CIBC$, $IIBC$, $OOBC$) and feature-driven cohesion (LCC) (3), where pla is a given PLA design, c is the number of components, and f is the number of its features. More details of the equations of the objective functions and their metrics are presented in Verdecia et al. (2017).

$$COE(pla) = \frac{1}{\sum_{i=1}^{c} H} \tag{1}$$

$$ACLASS(pla) = \sum_{i=1}^{c} CDepIn + \sum_{i=1}^{c} CDepOut \tag{2}$$

$$FM(pla) = \sum_{i=1}^{c} LCC + \sum_{i=1}^{f} CDAC + \sum_{i=1}^{f} CDAI + \sum_{i=1}^{f} CDAO + \sum_{i=1}^{f} CIBC + \sum_{i=1}^{f} IIBC + \sum_{i=1}^{f} OOBC \tag{3}$$

[5]The value of H is inverted in COE, as we are interested in maximizing the cohesion and MOA4PLA minimizes all the objective functions.

As evidenced in previous work (Choma Neto et al. 2019; Silva et al. 2020), these three objective functions are conflicting. COE and ACLASS are inversely proportional, whereas the competing between FM and COE (or ACLASS) is stronger or weaker depending on the PLA design given as input. The results of the evaluation of this work corroborates such statements.

3.4 Tool to Support the Application of MOA4PLA

To motivate practical adoption, easier replication of experiments, and new studies on PLA optimization, MOA4PLA is implemented in an open source tool named OPLA-Tool[6] (Féderle et al. 2015; Freire et al. 2020). OPLA-Tool allows users to choose different multi-objective algorithms, such as NSGA-II (Deb et al. 2002) and PAES (Knowles and Corne 2000), as well as an easy way to include new algorithms.

4 Proposal of Crossover Operators for PLA Design

In this section, we present methods adopted to keep consistency of PLAs generated during the evolutionary process and three crossover operators to intensify the PLA design optimization. It is important to mention that the methods proposed in Section 4.1 are intended for the crossover operation and not mutation, since the MOA4PLA approach checks whether the solution is valid or not, discarding the invalid ones from the population. This validation operation already existed and was maintained.

4.1 Consistency of the Generated Solutions

Software architectures are complex artifacts. In addition, when dealing with architectures using UML class diagrams, these artifacts are constrained to many detailed properties. These characteristics make any automatic strategy to deal with this artifact a difficult task. For instance, Harman and Tratt (2007) stated that it is hard to guarantee the accuracy of architectures when applying crossover operators. This was observed in a previous study, in which incomplete solutions were obtained during the *Feature-driven Crossover* application (Colanzi and Vergilio 2016). These incomplete solutions are due to the fact that attributes or methods, associated with features that were tangled in the same class, end up being lost after the crossover. Taking this into account, we propose two methods used to maintain the consistency and completeness when dealing with PLAs. These methods are *Diff* and *RemoveDuplicate*.

The *Diff* method is designed to search in the parents the elements missing in children, to consequently include these elements in the new generated solutions. Algorithm 1 presents the pseudocode of the *Diff* method. The list *diffListsElements*, with all elements of the parent that do not exist in the *offspring*, is created (line 1). For each element of *diffListElements*, it is checked if the child has the counterpart corresponding to *element* (line 3). If so, methods and attributes from *element* are included to the corresponding element in the child (line 5). Otherwise, the entire element is included in the child.

[6] https://github.com/otimizes/OPLA-Tool

Algorithm 1 Diff method.

```
   Input: parent, offspring
   Output: offspring
1  diffListElements ← all elements from parent that do not exist in offspring;
2  for each element ∈ diffListElements do
3  |  elementFromOffspring ← offspring.findElement(element);
4  |  if elementFromOffspring <> null then
5  |  |  elementFromOffspring.addMethodsAndAttributes(element);
6  |  |  else  offspring.addElement(element) ;
7  |  end if
8  end for
```

After the application of the *Diff* method, the *RemoveDuplicate* method is applied to remove duplicate elements (methods and attributes) from the child. These duplicate elements can be originated during the crossover operation. The decision of which element should be removed is based on the feature modularization, which means the element with the worst feature modularization is removed. Algorithm 2 presents the pseudocode of *RemoveDuplicate*. In lines 1 and 2, the list of elements and the list of the child's elements, respectively *listElems* and *elementsOffspring*, are initialized. *listElems* is an empty list and *elementsOffspring* contains all child's classes and interfaces. Then, the algorithm checks for all elements of the child whether *listElems* contains these methods or attributes (lines 3 to 6). When contained, it means that the method or attribute is duplicated. Next, the *getWorstMA* method (line 7) compares the current method or attribute *ma* to that previously one stored in *listElems*. This comparison takes into account the feature modularization of the elements (class or interface) to which the duplicate method or attribute belongs. The one with the worst modularization is selected to be removed from the *offspring* (line 8). The modularization of each element is evaluated based on feature interlacing and feature-driven cohesion. If the method or attribute is not in *listElems*, it is added to the list (line 10).

Algorithm 2 RemoveDuplicate method.

```
   Input: offspring
   Output: offspring
1  listElems ← empty List;
2  elementsOffspring ← offspring.getAllElements();
3  for each element ∈ elementsOffspring do
4  |  MAElements ← element.getAllMethodsAndAttributes();
5  |  for each ma ∈ MAElement do
6  |  |  if ma ∈ listElems then
7  |  |  |  worstMA ← offspring.getWorstMA(ma, lstElems[ma]);
8  |  |  |  offspring.removeMA(worstMA)
9  |  |  |  else  listElems.add(ma) ;
10 |  |  end if
11 |  end for
12 end for
```

The application of both proposed methods leads to the generation of complete and consistent solutions. On one hand, the *Diff* method guarantees the inclusion of missing elements

and, on the other hand, the *removeDuplicate* method eliminates the duplicate elements in a solution.

4.2 Proposed Crossover Operators

In this section, we present details of the three crossover operators that are one of the contributions of this work. The *Feature-driven Crossover* operator is a new version of the operator originally proposed in Colanzi and Vergilio (2016). The *Complementary Crossover* operator was inspired in the Simple Complementary Crossover proposed by Räihä et al. (2010). The *Modular Crossover* operator was based on the crossover proposed by Harman and Hierons (2002).

The original versions of the two latter operators use different chromosome representations, which cannot be directly mapped to a PLA design. Hence, the *Modular Crossover* and the *Complementary Crossover* proposed for the PLA design context use a different individual encoding. Our crossover operators are based on the PLA design representation explained in Section 3.1 (see Fig. 1). In this sense, the crossover operators perform the transfer of architectural elements (such as packages, classes, interfaces, methods, attributes, relationships, and variabilities) from parents to the children, following different rationales as presented in the next sections.

4.2.1 Feature-Driven Crossover

The *Feature-driven Crossover* operator is designed to improve the feature modularization of a PLA design. Its motivation is the fact that a PLA with low feature modularization is likely to suffer early modifications to accommodate new products. These modifications make it difficult to maintain the design stability over time. Figure 2 illustrates the application of this operator. On the left side of the figure, *Parent1* is represented by gray elements and *Parent2* by white elements. The features that are realized by each element, only of type *class* in this example, are represented by stereotypes, e.g., $<<F_n>>$. Let us suppose that the feature $<<F1>>$ was chosen to be swapped between the parents. Then, the elements of *Parent1* and *Parent2* that are not associated with $<<F1>>$ are copied to *Child1* and *Child2*, respectively. Next, the operator reallocates the elements of *Parent1* associated with $<<F1>>$ to *Child2* and the elements of *Parent2* to *Child1*. The resulting children are presented on the right side of the figure.

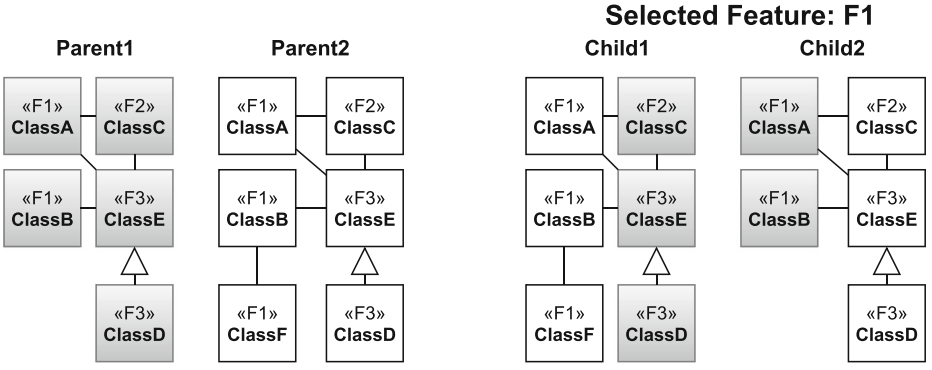

Fig. 2 Feature-driven Crossover

Algorithm 3 presents the pseudocode of the *Feature-driven Crossover* operator. Firstly, a list of all features of *parent1* is added in *F* (line 1) and a feature *fx* is randomly selected from *F* (line 2). After, all elements associated with *fx* of both parents are collected and placed in *c1* and *c2*. The two children are initialized as copies of *parent1* and *parent2* (lines 5 and 8). Next, all elements of *offspring1* and *offspring2* associated with the *fx* feature are removed (lines 6 and 9). Then, all elements and relationships associated with *fx* and that belong to *parent2* are added in *offspring1* (line 7). Similarly, all elements and relationships of *parent1* associated with *fx* are added to *offspring2* (line 10). The operations to guarantee the consistency of the solution are highlighted in lines 11–16. The *Diff* method is applied in both *offspring1* and *offspring2* to recover the elements potentially lost during the crossover operation (lines 11–12 and 15–17). The *RemoveDuplicate* method is applied to remove duplicate elements in *offspring1* and *offspring2* (lines 13 and 16). Hence, the *Diff* and *RemoveDuplicate* methods solve the problem of consistency of the original version of *Feature-driven Crossover* (Colanzi and Vergilio 2016). Finally, the links related to variability of each child are updated (lines 17–18).

Algorithm 3 Feature-driven crossover.

```
   Input: parent1, parent2
   Output: offspring1, offspring2
1  F ← parent1.getAllFeatures();
2  fx ← a randomly selected feature from F;
3  c1 ← parent1.getGetAllElementsAssociatedWithFeature(fx);
4  c2 ← parent2.getGetAllElementsAssociatedWithFeature(fx);
5  offspring1 ← new Solution(parent1);
6  offspring1.removeElementsRealizingFeature(c1,fx);
7  offspring1.addElementsRealizingFeature(c2,fx);
8  offspring2 ← new Solution(parent2);
9  offspring2.removeElementsRealizingFeature(c2,fx);
10 offspring2.addElementsRealizingFeature(c1,fx);
11 diff (parent1, offspring1);
12 diff (parent2, offspring1);
13 removeDuplicate (offspring1);
14 diff (parent1, offspring2);
15 diff (parent2, offspring2);
16 removeDuplicate (offspring2);
17 offspring1.updateVariabilities();
18 offspring2.updateVariabilities();
```

4.2.2 Complementary Crossover

The *Complementary Crossover* operator has the goal of creating offspring from parents that are complementary regarding given quality attributes. The original version of the *Complementary Crossover* (Räihä et al. 2010) is concerned with only two objectives, namely modifiability and reusability. On the other hand, MOA4PLA is a multi-objective approach that allows optimizing more than two objectives. Hence, we had to adapt our version of this operator to deal with a greater number of objectives. The first step for the application of this operator is to create a list with all individuals for each objective. Then, each list is sorted to rank the individuals according to a specific objective, i.e., sort the individuals in ascending

order based on an exclusive objective fitness. This enables the algorithm to select parents giving priority to the best solutions in relation to the objective related to the list. That is, the better the solution, the greater the chance it will be chosen to be selected as a parent. In MOA4PLA, the number of lists is similar to the number of objectives, which is defined by the user. Each list stores information related to a particular objective. The next step is to randomly select two lists, i.e., two objective functions, to perform the crossover. Then, one parent is picked from each list by applying the standard roulette wheel selection, preferably selecting the best ones for each objective.

Figure 3 presents an overview of the application of the *Complementary Crossover* operator. To avoid duplication of elements when applying the operator, the crossover point is defined only at one parent. In this illustrative example, the crossover point (CP) is defined for *Parent1*, on the left side of the figure. The crossover will operate in the parts of the solution after the crossover point, which is the hachured part in *Parent1*. *Parent2* is represented in gray, in the middle of the figure. To perform the crossover, firstly, the child on the right side, receives the part that precedes the crossover point of *Parent1*, illustrated as the white part in the child. Secondly, it receives elements from *Parent2* that are not yet in the child, represented by the gray part in the child. Finally, the child receives the part after the crossover point, the hachured part, that is not yet in the child. In this way, the completeness and consistency of the generated solution is guaranteed.

For dealing with PLAs representation, the definition of only one single crossover point is not suitable to apply the Complementary Crossover operator. To overcome this problem, three lists are created to different types of architectural elements, namely packages, classes, and interfaces; as the elements are organized in different lists. Thus, we can create crossover points for each one of them, that makes it possible to divide the PLA design into parts.

Algorithm 4 presents the pseudocode of our *Complementary Crossover* operator. Firstly, *chosenParent* receives one of the two parents that is randomly selected (line 2). Then, a crossover point *cp* is defined randomly for the *chosenParent* (line 3). After, the child receives all elements of *chosenParent* placed before the crossover point (line 4) and elements from the other parent that are not yet in the child (line 5). Next, the *chosenParent* elements that the child does not have yet, placed after the crossover point, are added to the child (line 6). Finally, the operations performed in the lines 5 and 6 are the *Diff* method, adding elements from parents that aren't in the child, and variability links of the $offspring$ are updated (line 7). As only elements that the child does not have are added, there is no possibility of duplicate methods and attributes. Therefore, it is not necessary to apply the *removeDuplicate* method.

Algorithm 4 Complementary crossover.

Input: *parent1*, *parent2*
Output: *offspring*
1 *offspring* ← new Solution;
2 *chosenParent* ← a randomly selected parent;
3 *cp* ← a randomly selected point from *chosenParent*;
4 *offspring* ← all elements of *chosenParent* that are before *cp*;
5 diff (*otherParent*, *offspring*);
6 diff (*chosenParent*, *offspring*);
7 *offspring*.updateVariabilities();

Fig. 3 Complementary Crossover

4.2.3 Modular Crossover

The *Modular Crossover* operator focus on preserving allocations of complete or partial modules from a parent to a child. Differently from the Feature-driven Crossover, which is based on grouping architectural elements of a specific feature, the Modular Crossover relies on structural aspects of the PLA modules, which are packages or components. The rationale is that a module in a PLA may already be well modularized, then, the child will likely benefit from having this package as it is. In summary, the Modular Crossover guarantees the full preservation of at least one module of one parent in the child and the partial preservation of the other modules (Harman and Hierons 2002).

We adapted the original Modular Crossover (Harman and Hierons 2002) operator to optimize PLAs. Some elements of the selected module of a PLA can relate to one or more elements of one or more modules. This implies the need to also copy other modules related to the chosen one to the child. Additionally, instead of excluding the modules from both parents as in the original version of this operator (Harman and Hierons 2002), our adaptation flags the modules as "deleted", in order to not be selected again when completing the children. Thus, every time a module with relationships is included to the child, it should be checked whether the target module that the chosen module relates to is marked as "deleted". If so, the module is not included to the child, otherwise, it is included.

Two constraints are taken into account regarding the relationships of the modules. Figure 4 presents an illustrative example to explain such constraints. Firstly, only relationships that imply that the source module depends on the target module will be taken into

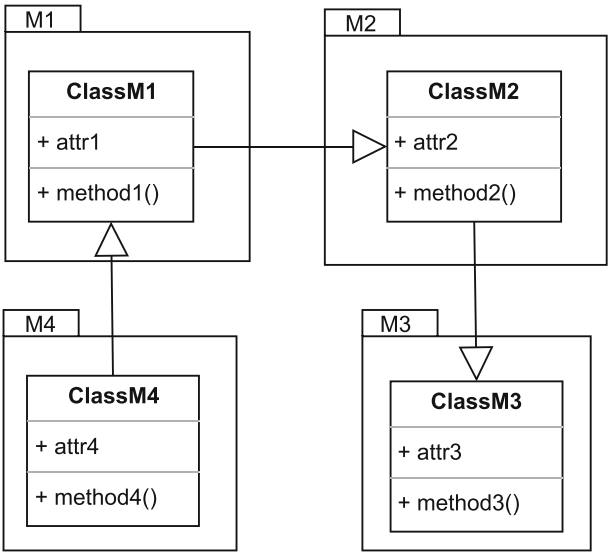

Fig. 4 Dependency between modules

account. For example, if the module *M1* has a relationship with the module *M4*, but does not depend on *M4*, only *M1* is preserved in the child. Secondly, exclusion of indirect dependencies. For example, if *M1*, the module chosen, depends on *M2*, and *M2* depends on *M3*, only *M1* and *M2* will be preserved in the child, as they are directly related. These constraints are important because a well-designed PLA mostly have modules/components taking part in relationships to other ones. That is, if the operator maintains all relationships, i.e., direct and indirect relations, the child would tend to be very similar to one of the parents.

The last adaptation of the Modular Crossover operator is related to the selection of the module from one of the parents, which in our version takes into account the particularities of PLAs, such as variation points and SPL features. We opt for prioritizing the selection of module with the highest number of variation points. In case of a tie, the criterion changes to the module with the highest number of relationships. If this condition also results in a tie, the tiebreaker criterion is the smallest number of features.

Algorithm 5 presents the pseudocode of the Modular Crossover. The lines 12 to 32, are related to the criteria for selecting the module to be preserved to the child, which is applied using *listPackage*. The elements of the module selected from *listPackage* are included to the child (line 28). After that, the existing elements that were passed to the child are flagged as "deleted" from both parents (lines 34 and 35, respectively). With the completion of the while loop (line 36), the method *addElementsNotInPackage* completes the child (lines 37 and 38). Then, the *Diff* (lines 39 and 40) and *RemoveDuplicate* (line 41) methods are applied to maintain the consistency of the solution. In line 42, the references links of variabilities are updated.

5 Empirical Study Design

This section describes details of the empirical study conducted to evaluate the crossover operators proposed in this work.

5.1 Research Questions

In our study, we aim at answering the following research questions (RQ):

RQ1—What is the benefit of using crossover operators in the optimization of PLA design in comparison to the state of the art? This is the main RQ of our study because after proposing the three crossover operators, it is important to know whether they are beneficial for search-based PLA design or not. Benefit in this context means the positive impact on the characteristics of solutions that result in a well-designed PLA. These characteristics are measure based on architectural properties such as feature modularization, relational cohesion or class coupling, which are directly related to the improvement of the fitness values of these solutions. In this question, we intend to characterize which are the main benefits of the proposed crossover operators for PLA design.

RQ1.1—What is the performance of the proposed crossover operators for PLA design optimization in terms of quality indicators? To answer this question, we conducted a quantitative analysis, relying on quality indicators adopted in multi-objective optimization, in order to evaluate the obtained results. The PLAs used to answer this question were obtained by selecting the solutions among those that presented the best trade-off after merging all the solutions from the 30 independent runs of each experimental configuration.

Algorithm 5 Modular crossover.

```
Input: parent1, parent2
Output: offspring
offspring ← new Solution;
while at least one parents has components do
    package1 ← random module from a random parent;
    package2 ← package1 corresponding in other parent if exists;
    nRelationship1 ← package1.getRelationships.size();
    nVariantPoints1 ← package1.getVariantPoints.size();
    nFeatures1 ← package1.getFeatures.size();
    nRelationship2 ← package2.getRelationships.size();
    nVariantPoints2 ← package2.getVariantPoints.size();
    nFeatures2 ← package2.getFeatures.size();
    listPackage ← null;
    if nVariantPoint1 == nVariantPoint2 then
        if nRelationship1 == nRelationship2 then
            if nFeature1 ≤ nFeature2 then
                listPackage ← chosenParent.getRelatedPackages(package1);
            else
                listPackage ← otherParent.getRelatedPackages(package2)
            end if
        else
            if nRelationship1 < nRelationship2 then
                listPackage ← otherParent.getRelatedPackages(package2);
            else
                listPackage ← chosenParent.getRelatedPackages(package1)
            end if
        end if
    else
        if nVariantpoint1 < nVariantpoint2 then
            listPackage ← otherParent.getRelatedPackages(package2);
        else
            listPackage ← chosenParent.getRelatedPackages(package1)
        end if
    end if
    offspring.addElements(listPackage);
    chosenParent.removeElements(listPackage);
    otherParent.removeElements(listPackage);
end while
offspring.addElementsNotInPackage(parent1);
offspring.addElementsNotInPackage(parent2);
Diff(parent1, offspring);
Diff(parent2, offspring1);
RemoveDuplicate(offspring);
offspring.updateVariabilities();
```

RQ1.2—What is the impact of the crossover operators on feature modularization of the PLA design solutions? Taking into account the importance of feature modularization for SPL engineering, it is important to know which is the impact caused by the crossover operation on this architectural property. To answer this RQ, we analyzed the quantitative data and performed a detailed analysis on several of the obtained design solutions. The PLA solutions used to answer this question were obtained by selecting those that presented the best trade-off after merging all the solutions from the 30 independent runs. Furthermore, for the detailed analysis of the obtained solutions, those with the best ED value regarding the 30 independent runs were used.

RQ1.3—Which crossover operator generates better solutions according to the experts? To answer this question, we invite software engineer experts in SPL engineering for evaluating eight solutions generated in our empirical study. So, this was an impartial evaluation performed by people who do not know our proposed operators. For the analysis with the experts, the solutions with the best ED and trade-off value of each experiment were selected, thus obtaining eight solutions, i.e., one solution per experiment.

5.2 Quantitative Study

A quantitative study was conducted to evaluate the performance of the proposed crossover operators. We present details of this study next.

5.2.1 Subject PLA Designs

In the experimental study we used three PLA designs: Arcade Game Maker (AGM) (2009), Mobile Media (MM) (Contieri et al. 2011), and a real-world SPL named Electronic Tickets in Urban Transportation (BET) (Donegan and Masiero 2007).[7] AGM was created by the Software Engineering Institute (SEI). This SPL is composed of three arcade games, namely Brickles, Bowling and Pong (2009). MM is a mobile application that implements features to handle with music, video, and photo in portable devices (Young 2005). BET was developed for the management of municipal public transport bus services, offering various features for passengers and road companies, such as using an electronic card for transportation payment, automatic toll opening, and unified travel payment (Donegan and Masiero 2007). Table 1 presents architectural elements numbers of both PLA designs.

5.2.2 Experimental Configurations

For investigating the behavior of every single crossover, as well as their combinations, our study considered eight experimental configurations, as follows:

1. **BASE** is the baseline experiment where only the six mutation operators of MOA4PLA are applied, as done in previous studies (Colanzi et al. 2014; Choma Neto et al. 2019).
2. **FdC** applies the *Feature-driven Crossover* and the six mutation operators.
3. **CC** applies the *Complementary Crossover* and the six mutation operators.
4. **MC** applies the *Modular Crossover* and the six mutation operators.
5. **FdC+CC** applies the *Feature-driven Crossover*, the *Complementary Crossover*, and the six mutation operators.

[7] The file of each PLA used to conduct the study is available at https://github.com/otimizes/OPLA-Tool/tree/master/plas

Table 1 Number of architectural elements of the subject PLA designs

PLA	Components	Interfaces	Classes	Mandatory features	Variable features
AGM	9	14	30	6	5
MM	8	15	14	7	7
BET	56	30	115	8	10

6. **FdC+MC** applies the *Feature-driven Crossover*, the *Modular Crossover*, and the six mutation operators.
7. **CC+MC** applies the *Complementary Crossover*, the *Modular Crossover*, and the six mutation operators.
8. **All** applies *all crossover operators* and the six mutation operators.

5.2.3 Search-Based Algorithm, Objective functions, and Parameter Settings

All experiments were executed using NSGA-II (Deb et al. 2002) implemented on top of OPLA-Tool v2.0 (Freire et al. 2020). We chose NSGA-II because it has been successfully used in many previous studies (Colanzi et al. 2014, 2020; Colanzi and Vergilio 2016) and has good performance to optimize three competing objectives (as can be observed in Fig. 5), allowing a wide exploration in the search space (Fig. 8). The objective functions were the same for all experiments: COE, ACLASS and FM (presented in Section 3), similarly to Choma Neto et al. (2019). The parameters of NSGA-II were set up as described next.

In our previous study (Silva et al. 2020), we performed an experimental calibration[8] of parameters to define the best configuration of crossover and mutation rates for each experiment. We adopted the same setting to execute the experiments of this extension. The setting for all experimental configurations used in the present work was crossover rate of 0.4 and mutation rate of 0.8. Population size of 100 individuals and 300 generations. These same parameters were used for 30 independent runs of each experimental configuration.

Regarding the execution time, each run spent around 5 h for the smallest PLA designs (AGM and MM) and up to 24 h for the largest PLA (BET). In real PLAs, such as BET, an architect will probably need several days to manually design and evaluate alternative PLAs. Hence, a runtime of 24 h required to automatically obtain some alternative designs for a PLA might not be obstructive for practical use. There is not a significant difference in the runtime when applying a particular crossover operator. However, this analysis is out of the scope of this study.

5.2.4 Quality Indicators and Statistical Tests

To support the result analysis and computing the quality indicator, we composed three sets of solutions, namely, PF_{approx}, PF_{known} and PF_{true}. PF_{approx} is the Pareto front of non-dominated solutions obtained in each run of an experiment. As we run each experiment by 30 times, we have 30 PF_{approx} sets for each experiment. PF_{known} is the set of non-dominated solutions found by an experiment, considering the union of all solutions obtained in all its runs, eliminating the dominated ones. PF_{true} is conceptually known as the set with

[8]Results of the experimental calibration are available at https://doi.org/10.5281/zenodo.6516279

ideal solutions for a problem. As the PF_{true} of our problem is not known in advance, thus, we adopted a common way to estimate this Pareto front that is using the non-dominated solutions found by all experiments in all runs (Zitzler et al. 2002).

The quality indicators adopted to compare the experiment results are Hypervolume, Coverage, and Euclidean Distance to the Ideal Solution, which evaluate spread and convergence of solutions (Li and Yao 2019).

Hypervolume (H) is a quality indicator that measures the n-dimensional volume between a Pareto front and a specific reference point (Zitzler et al. 2003). In our experiments, which are minimization optimizations and we chose as reference point the worst values of the objective functions, the higher the hypervolume value is, the greater the coverage area, reflecting a better front. To compute H we normalized each PF_{approx} between 0 and 1, and adopted a reference point with the value of 1.01 for all objectives, which is higher than the worst possible value. Hence, the reference point used in this paper was (COE = 1.01, ACLASS = 1.01, FM = 1.01).

Coverage (C) measures the dominance between two sets of non-dominated solutions (Zitzler and Thiele 1998). C(PFa, PFb) returns a value between 0 and 1 according to how much the set PFb is dominated by set PFa. Similarly, but in an opposite way, C(PFb, PFa) returns how much PFa is dominated by PFb. On one hand, C equal to 0 indicates that the solutions of the former set do not dominate any element of the latter set; on the other hand, value equal to 1 indicates that all elements of the latter set are dominated by elements of the former set. In our study, we used PF_{known} of each experiment to compute C.

Another quality indicator adopted is the *Euclidean Distance to the Ideal Solution (ED)*. The ED value is a distance measure that designates which solution is closest to an "ideal solution". For minimization problems, the ideal solution is the one that contemplates the lowest value possible for the objective function being optimized (Zeleny and Cochrane 1973). The ideal solution adopted to be used as reference point to the *ED* value was (COE = 0, ACLASS = 0, and FM = 0), as this work used three objective functions in a minimization problem. The PF_{known} sets were used to find the solution with the lowest ED of each experiment.

In addition to compute the quality indicators, to our analysis we also rely on statistical tests to an in-depth discussion. *Shapiro-Wilk* statistical test (Surhone et al. 2010) was used to investigate if the sample sets have normal distribution, which is the basis for deciding the statistical methods for data analysis (Mishra et al. 2019). In order to statistically compare the results found by hypervolume, *Kruskal-Wallis Pairwise* test was applied since there are eight independent data sets (one for each experimental configuration) with non-normal distribution, also having independent variables (Juristo and Moreno 2013). The *Kruskal-Wallis* test aims to determine if there is an overall difference among the various sampling groups, which is commonly needed in SBSE (Colanzi et al. 2020). However, the test does not specify where the difference lies between groups (Kruskal and Wallis 1952). We adopt a confidence of 95% (p-value $\leq$0.05) in order to verify the statistical difference between the sample sets. To further analysis, we also compute the effect size with the *Vargha-Delaney's* $\hat{A}_{12}$ measure (Vargha and Delaney 2000).

5.3 Qualitative Study

As important as the quantitative analysis with quality indicators, is to know the opinion of experts about the obtained solutions. With this in mind, this qualitative analysis aims to complement the quantitative one.

5.3.1 Expert Characterization

The qualitative study was performed with five software engineers, who are experts in SPL engineering, what is important to evaluate the alternatives of PLA design obtained during the optimization process. The sampling of the participants was done based on convenience. Tables 2 and 3 present the characterization of the participants. Most of them are masters in Computer Science and PhD Candidates. Three of them also have positions in the industrial sector. The entire set of participants have experience in their respective acting sector, ranging from 5 to 25 years. Their experience in software development varies from 4 to 18 years. Three participants have moderate experience with UML modeling, whereas two have advanced experience (Table 3). Four of them have moderate experience with SPL engineering and variability management and the remaining participant has advanced experience with this subject. They are from three different Brazilian cities and their participation in the study was remote due to the COVID-19 pandemic.

5.3.2 Qualitative Study Design

For conducting the qualitative study, a systematic guideline was followed with a series of systematized steps, presented by the following protocol:

1 Application of the consent form and questionnaire to characterize the profile of the participants;
2 Providing a video about SPL;
3 Providing a PDF file containing information about SPL properties;
4 Providing a PDF file containing specific information on the AGM SPL;
5 Providing 4 PLAs of AGM for the evaluation;
6 Application of a questionnaire about the evaluated PLAs;
7 Providing a PDF file containing specific information on the MM SPL;
8 Providing 4 PLAs of MM for the evaluation; and,
9 Application of a questionnaire about the evaluated PLAs.

For conducting the qualitative analysis, we decided to use only solutions of AGM and MM aiming at avoiding fatigue of the participants, because both SPLs are smaller than BET. For each SPL, four alternatives of PLA design were selected to be evaluated by the experts. These alternative PLA designs were evaluated by using a questionnaire applied

Table 2 Participant characterization

Participant	Education level	Sector of activity	Experience in the sector of activity	Experience in software development
P1	PhD Candidate	Academic	8 years	4 years
P2	PhD Candidate	Academic	5 years	8 years
P3	PhD Candidate	Academic/ Industrial	10 years	8 years
P4	PhD Candidate	Academic/ Industrial	25 years	18 years
P5	Masters	Industrial	15 years	10 years

Table 3 Participant characterization regarding UML and SPL engineering

Participant	UML modeling experience	SPL engineering experience
P1	Moderate	Moderate
P2	Moderate	Moderate
P3	Advanced	Advanced
P4	Advanced	Moderate
P5	Moderate	Moderate

using Google Forms.[9] Such a questionnaire encompasses six essay questions related to architectural properties (objectives) optimized in the search and the preferred solution by the experts. The questionnaire was carried out with the participants individually, on predefined dates and times, following each participant's availability schedule. The questions are:

- *Q1*: *What is your opinion about the relational cohesion of this PLA design?*
- *Q2*: *What is your opinion about the coupling of this PLA design?*
- *Q3*: *What is your opinion about the interface size of this PLA design?*
- *Q4*: *What is your opinion about the feature modularization of this PLA design?*
- *Q5*: *In our opinion, which is the best alternative of PLA design regarding feature modularization?*
- *Q6*: *In our opinion, which are the most important architectural properties to be evaluated in a PLA design?*

For each SPL, questions *Q1, Q2, Q3* and *Q4* were answered four times for each alternative of PLA design. Question *Q5* was answered once for each SPL, asking the participant to choose the best of the four alternatives under his point of view. Question *Q6* was answered once at the end of the questionnaire. Before answering the questionnaire, the participants received a training that includes a video and a document about SPL engineering. In addition, we provided documents containing the specification of requirements, features and variabilities of AGM and MM. It is important to note that participants were guided during the entire process.

We selected the solution with the lowest ED, i.e., with the best trade-off among the objectives, obtained by the experiments FdC+CC, FdC+MC, and All. We choose only these experimental configurations because they achieved solutions with the best fitness values and the lowest ED for AGM and MM (see Table 8), as we discuss in the quantitative analysis. The best solution of BASE for AGM and MM was also selected in order to compare the solutions generated using crossover operators with the baseline. Hence, it is possible to analyze the impact caused by the proposed crossover operators in the obtained solutions.

The PLA designs were provided to each participant. Each design was numbered with a unique identifier to hide the name of the experiment that generated it. The PLA designs were identified as AGM-1, AGM-2, AGM-3, AGM-4, MM-1, MM-2, MM-3, and MM-4. No further information about the solutions was provided to prevent any bias. So, the participants did not know which experiment generated which solution, leading to an impartial evaluation. The four designs involved in this study and their respective experiments are: BASE (AGM-1 and MM-1), FdC+CC (AGM-2 and MM-2), FdC+MC (AGM-3 and MM-3) and All (AGM-4 and MM-4). Recall that BASE is the baseline experiment, where solutions

[9] https://www.google.com/forms/about/

are generated without crossover. FdC+CC applies the *Feature-driven Crossover* and *Complementary Crossover* operators, whereas FdC+MC applies *Feature-driven Crossover* and *Modular Crossover*. Finally, All uses the three proposed crossover operators: *Feature-driven Crossover*, *Complementary Crossover* and *Modular Crossover*. The participants spent an average of 2 h evaluating the PLA design alternatives.

6 Empirical Study Results

In this section, we present the quantitative and qualitative analysis of the results and answer the posed research questions. Our goal is to identify the impact of the crossover operators proposed in this work (*Complementary Crossover, Feature-driven Crossover* and *Modular Crossover*) on the PLA design solutions when compared to the state-of-the-art baseline. The figures used in this section illustrate excerpts extracted from the PLA design solutions that contain only the elements of interest for the analysis.[10] In these figures, the features are assigned to architectural elements using UML stereotypes.

6.1 Quantitative Analysis

All the data from figures and tables presented in this section take into account the 30 independent runs of each experiment. Figure 5 presents the surface covered by the solutions found for AGM, MM and BET in the search space. For AGM (Fig. 5(a)), FdC+CC, FdC+MC and All were better when considering the objective function related to feature modularization (FM), covering the most of the search space below the other experiment fronts. FdC+MC, in turn, was better in terms of class coupling, i.e., the ACLASS objective function. When analyzing relational cohesion, i.e., the COE objective function, All presented better surface values. CC+MC solutions are concentrated in the middle of the other fronts. It is important to note that B is spread on an extensive part of the search space above the other experiments, being close to the surfaces of the experiments that use a single crossover operator (FdC, CC and MC). Thus, for AGM, the experiments that use combinations of the crossover operators, cover the search space below the others, presenting better results in general.

Similarly to AGM, FdC+MC, FdC+CC and All showed better distribution for MM when analyzing the FM, COE, and ACLASS objective functions in an integrated manner. The experiment All presented short peak moments for FM values. As we can see, the surface of FdC+MC covers the search space below the other fronts to a large extent, more comprehensively when compared to the AGM graph. For MM, the experiments that combine crossover operators also show a better layout of the surfaces.

Differently from AGM and MM, FdC+MC, CC+MC, CC and MC achieved similar distribution of solutions on search space for BET, as can be seen in Fig. 5(c). FdC+CC is also very similar to this group, except for the peak FM values in a given region of its surface. The baseline experiment (B) presents the highest peaks in the distribution of its surface. Taking into account the three objectives should be minimized, it can be noticed that FdC+MC, FdC+CC and All generated solutions with better distribution than the other experiments.

[10] The designs used in the qualitative analysis are available at https://doi.org/10.5281/zenodo.6516279.

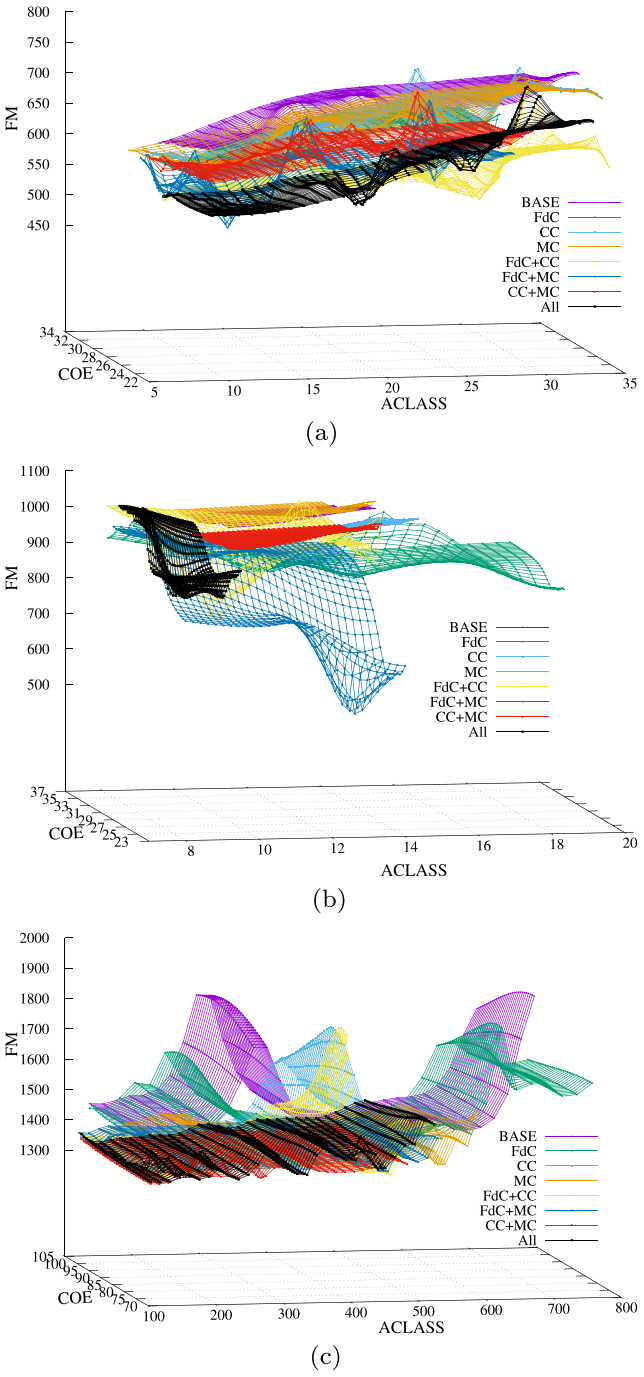

Fig. 5 Search space covered by solutions of AGM, MM and BET

Coverage The results of coverage for AGM, presented in Table 4, showed that the solutions of FdC+CC, FdC+MC and All dominate all the solutions found by B, FdC, CC and MC. CC+MC solutions dominate 67%, 80% and 54% of the FdC, CC and MC solutions, respectively. The solutions generated by FdC+CC, FdC+MC and All are not dominated by the solutions of FdC, CC, MC, while the solutions of FdC and CC dominate 27% of CC+MC solutions, a relatively small percentage. This information reveals that all the solutions generated by the experiments that use combinations of crossover operators, except for CC+MC, dominate the solutions of the experiments that apply a single crossover operator.

Performing an analysis among the experiments that combine crossover operators, FdC+MC shows better results for coverage, since the FdC+MC solutions dominate 67%, 100% and 67% of the solutions achieved by FdC+CC, CC+MC and All, respectively.

The coverage results for MM, presented in Table 5, are slightly different from AGM results, because the experiments that combine crossover operators (FdC+CC, FdC+MC, All) do not dominate the entire set of solutions achieved by the experiments that apply a single crossover operator. FdC+CC solutions dominate 66%, 90% and 100% of the solutions generated by FdC, CC and MC, respectively. The FdC+MC solutions dominate 87%, 80% and 100% of the FdC, CC and MC solutions, respectively. Regarding All, its solutions dominate 71% of FdC solutions, and all its solutions dominate the CC and MC solutions. MC solutions dominate only solutions belonging to B (100% in total). Similar to what was found for AGM, FdC+MC obtained better coverage results, since its solutions dominate 69%, 71% and 50% of FdC+CC, CC+MC and All solutions, respectively.

Regarding the results of coverage for BET (Table 6), all solutions achieved by B are dominated by solutions of all other experiments. In addition, all solutions of FdC are dominated by the solutions of FdC+CC, FdC+MC, CC+MC and All. The original PLA design of BET has high feature modularization, what might justify the worse performance of Exp _FdC for this SPL. Interestingly, the solutions of CC dominate 91%, 49%, 67%, 63% and 10% of the FdC, MC, FdC+CC, FdC+MC and All solutions, respectively. However, the best experiment with respect to coverage was CC+MC, since its solutions dominate 98% and 88% of the CC and MC solutions, respectively. In addition, the CC+MC solutions dominate 96% of the FdC+CC solutions, 91% of the FdC+MC solutions and 93% of the All solutions.

Hypervolume Regarding the hypervolume indicator, Table 7 presents the results of the Kruskal-Wallis test (p-value) and the $\hat{A}_{12}$ measure (effect size), for the 30 independent runs

Table 4 Results of the coverage indicator for AGM

Experiment	AGM							
	B	FdC	CC	MC	FdC+CC	FdC+MC	CC+MC	All
B	-x-	0.00	0.00	0.00	0.00	0.00	0.00	0.00
FdC	1	-x-	0.28	0.14	0.00	0.00	0.27	0.00
CC	1	0.66	-x-	0.36	0.00	0.00	0.27	0.00
MC	1	0.67	0.68	-x-	0.00	0.00	0.36	0.00
FdC+CC	1	1	1	1	-x-	0.29	0.95	0.47
FdC+MC	1	1	1	1	0.67	-x-	1	0.67
CC+MC	1	0.67	0.80	0.54	0.00	0.00	-x-	0.67
All	1	1	1	1	0.44	0.18	0.91	-x-

Table 5 Results of the coverage indicator for MM

Experiment	MM							
	B	FdC	CC	MC	FdC+ CC	FdC+ MC	CC+ MC	All
B	-x-	0.00	0.00	0.33	0.00	0.00	0.00	0.00
FdC	1	-x-	0.90	1	0.46	0.17	0.71	0.50
CC	1	0.04	-x-	1	0.00	0.05	0.57	0.00
MC	1	0.00	0.00	-x-	0.00	0.00	0.00	0.00
FdC+CC	1	0.66	0.90	1	-x-	0.28	0.86	0.50
FdC+MC	1	0.87	0.80	1	0.69	-x-	0.71	0.50
CC+MC	1	0.12	0.50	1	0.07	0.11	-x-	0.00
All	1	0.71	1	1	0.38	0.28	1	-x-

of NSGA-II. The magnitude of the effect size is presented by using symbols, according to the caption note (*) in the table.

For AGM, the statistical test pointed to significant difference between 11 of experiments, in gray cells, with confidence of 95% (p-value ≤ 0.05). Among these, for 10 pairs have effect size of magnitude large (▲). However, none of the experiments has supremacy over the whole set of experiments, as can be seen in the boxplots presented in Fig. 6(a). We observe that All, FdC+CC, FdC and MC are better than B. Among all the experiments that apply crossover and mutation operators, FdC, FdC+CC and MC have the best performance with respect to hypervolume.

For MM, a great amount of the experiments (18 out 28 paired comparisons) also obtained statistical differences (Table 7). Regarding the effect size, we can observe large magnitude for all of those pairs of experiments. To reason about the better experiments, we can analyze the boxplot in Fig. 6(b). Interestingly, the boxplots show that the baseline (B), MC and CC obtained the best results, while FdC+MC had the worst result of hypervolume among the evaluated experiments. However, considering the best values obtained for the FM objective function, FdC+MC presents the best result, as can be seen in Figs. 7 (for MM) and 5(b). This

Table 6 Results of the coverage indicator for BET

Experiment	BET							
	B	FdC	CC	MC	FdC+ CC	FdC+ MC	CC+ MC	All
B	-x-	0.00	0.00	0.00	0.00	0.00	0.00	0.00
FdC	1	-x-	0.00	0.00	0.00	0.00	0.00	0.00
CC	1	0.91	-x-	0.49	0.67	0.63	0.00	0.10
MC	1	0.97	0.08	-x-	0.11	0.28	0.02	0.10
FdC+CC	1	1	0.2	0.59	-x-	0.68	0.00	0.04
FdC+MC	1	1	0.10	0.63	0.12	-x-	0.03	0.10
CC+MC	1	1	0.98	0.88	0.96	0.91	-x-	0.93
All	1	1	0.84	0.81	0.85	0.84	0.05	-x-

Table 7 Results for the statistical test and effect size of Hypervolume for the 30 independent runs of NSGA-II

Experiment	AGM		MM		BET	
	p-value	Effect size[a]	p-value	Effect size[a]	p-value	Effect size[a]
B vs FdC	2.80E-05	0.93 ▲	3.60E-07	0.00 ▲	7.32E-01	0.86 ▲
B vs CC	1.13E-01	0.53 ≃	9.88E-01	0.42 ≃	8.10E-14	1 ▲
B vs MC	3.31E-03	0.67 △	1.00E+00	0.54 ≃	6.60E-04	1 ▲
B vs FdC+CC	1.20E-06	0.97 ▲	4.60E-04	0.13 ▲	2.50E-12	1 ▲
B vs FdC+MC	6.67E-01	0.74 △	6.10E-14	0.00 ▲	4.84E-02	0.97 ▲
B vs CC+MC	9.95E-01	0.40 ≃	5.36E-01	0.31 △	8.10E-14	1 ▲
B vs All	5.29E-03	0.87 ▲	6.10E-06	0.12 ▲	1.20E-13	1 ▲
FdC vs CC	3.79E-01	0.43 ≃	4.20E-05	0.95 ▲	9.30E-14	1 ▲
FdC vs MC	9.64E-01	0.50 ≃	8.20E-08	1 ▲	1.61E-01	0.93 ▲
FdC vs FdC+CC	9.99E-01	0.55 ≃	8.61E-01	0.59 ≃	1.60E-07	1 ▲
FdC vs FdC+MC	3.43E-02	0.20 ▲	3.92E-02	0.07 ▲	8.52E-01	0.78 ▲
FdC vs CC+MC	4.60E-07	0.09 ▲	4.10E-03	0.90 ▲	4.80E-13	1 ▲
FdC vs All	9.36E-01	0.36 ≃	1.00E+00	0.41 ≃	3.30E-09	1 ▲
CC vs MC	9.59E-01	0.55 ≃	9.50E-01	0.60 ≃	1.10E-09	0.00 ▲
CC vs FdC&CC	1.11E-01	0.65 ≃	1.49E-02	0.24 ▲	1.75E-02	0.07 ▲
CC vs FdC+MC	9.75E-01	0.45 ≃	1.80E-13	0.01 ▲	2.50E-13	0.00 ▲
CC vs CC+MC	1.14E-02	0.23 ▲	9.69E-01	0.38 ≃	8.04E-01	0.20 ▲
CC vs All	9.79E-01	0.55 ≃	4.50E-04	0.18 ▲	1.09E-01	0.09 ▲
MC vs FdC+CC	7.21E-01	0.59 ≃	1.50E-04	0.11 ▲	2.62E-02	1 ▲
MC vs FdC+MC	4.20E-01	0.34 ≃	1.10E-13	0.00 ▲	9.40E-01	0.24 ▲
MC vs CC+MC	1.20E-04	0.18 ▲	3.71E-01	0.30 △	9.40E-06	1 ▲
MC vs All	1.00E+00	0.45 ≃	1.60E-06	0.11 ▲	2.88E-03	1 ▲
FdC+CC vs FdC+MC	4.57E-03	0.18 ▲	1.80E-04	0.05 ▲	2.70E-04	0.00 ▲
FdC+CC vs CC+MC	1.30E-08	0.09 ▲	2.54E-01	0.68 △	5.80E-01	0.83 ▲
FdC+CC vs All	6.40E-01	0.34 ≃	9.87E-01	0.42 ≃	9.99E-01	0.61 ≃
FdC+MC vs CC+MC	2.00E-01	0.30 △	1.70E-10	0.98 ▲	1.00E-08	1 ▲
FdC+MC vs All	5.03E-01	0.69 △	7.17E-03	0.93 ▲	1.30E-05	1 ▲
CC+MC vs All	2.20E-04	0.85 ▲	2.46E-02	0.22 ▲	9.20E-01	0.30 △

[a]Values in gray cells indicate statistical difference. The symbols for the magnitude of the effect size are: "≃" negligible, "▽" small magnitude, "△" a medium magnitude, and "▲" a large magnitude

peak in the FM value might have negatively influenced the calculation of the normalized hypervolume. Therefore, it is not possible to state that FdC+MC is the worst experiment for MM, as it presents superior results when looking solely for values of fitness of the solutions.

Analyzing BET, it is possible to notice that 19 out 28 paired comparisons between the experiments are statistically different, with large magnitude. When checking the boxplot (Fig. 6(c)), it is noticeable that all experiments that apply crossover operators obtained better values in terms of hypervolume than the baseline experiment. This evidences that the use of crossover is effective in this SPL with respect to hypervolume. Taking into account the

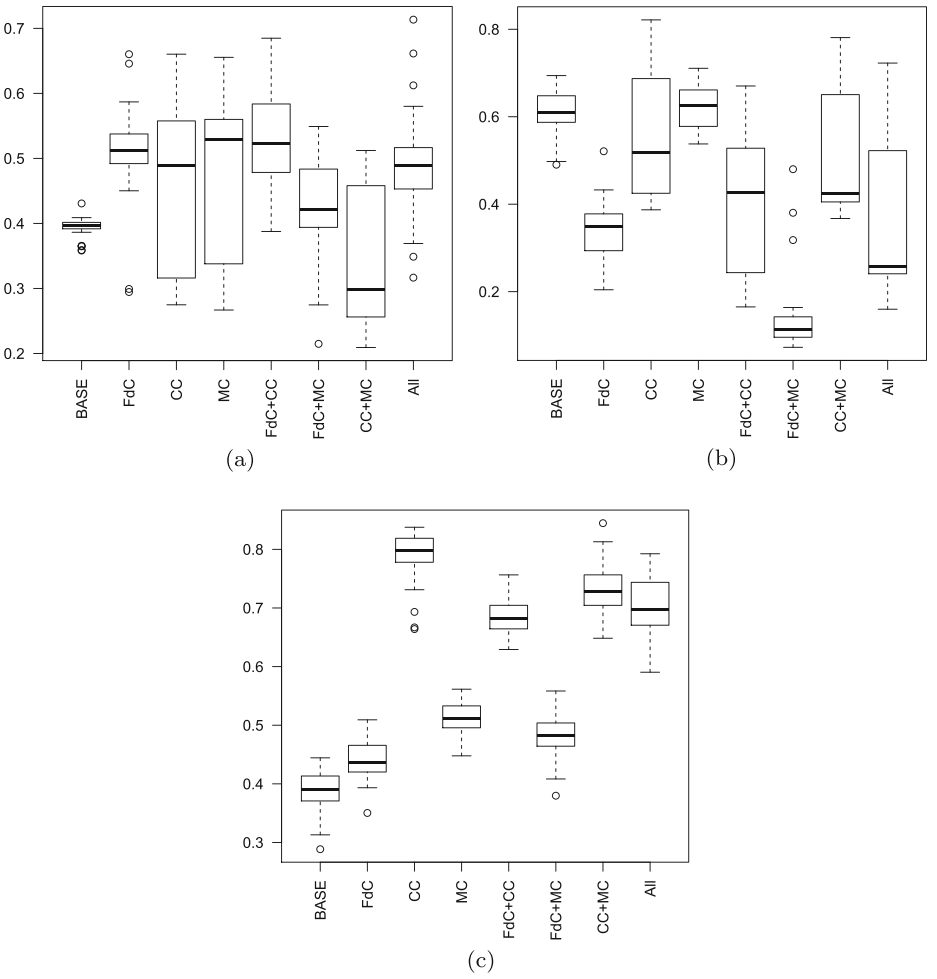

Fig. 6 Boxplot of the hypervolume

hypervolume values (Fig. 6(c)), the best performing experiments for BET are CC, FdC+CC, and CC+MC.

Feature Modularization Fitness Values In an additional analysis of the boxplots presented in Fig. 6, we can see that, for AGM and MM, the experiments B and FdC seem to behave inversely proportional, as also observed in our previous work (Silva et al. 2020). The *Feature-driven Crossover* operator is clearly dependent on the characteristics of the subject PLAs. We performed an in-depth analysis in the AGM and MM original designs and observed the features of AGM are better modularized than MM (Silva et al. 2020). The impact of feature modularization can be noticed in Fig. 6(a) and (b), where the experiments that apply the *Feature-driven Crossover* operator (FdC, FdC+CC, FdC+MC and All) have worse values of hypervolume.

Figure 7 presents the behavior of each experiment over generations in terms of FM fitness values. The slower convergence of FdC, FdC+CC, FdC+MC and All for MM corroborates our analysis that the original design of MM has worse feature modularization than AGM.

The experiments that use any crossover operator reached solutions with better FM values than the baseline experiment for the three SPLs. In addition, the experiments that apply more than one crossover operator (FdC+CC, FdC+MC, CC+MC and All) often reached the best values of FM, showing that the proposed crossover operators complement each other and that their joint application is more profitable.

Furthermore, in spite of FdC+MC did not obtain the best solutions in relation to the hypervolume, it is noted that it obtained the best FM values in two (AGM and MM) of the three SPLs used in the experiments. This shows that, in general, this combination of crossover operators generates interesting results in terms of feature modularization. It is also important to highlight the noticeably faster convergence of FM values with FdC+MC for MM compared to AGM (Fig. 7 for AGM and MM). Given the low feature modularization of the original PLA design of MM, there is a greater range of optimization in this PLA reached by the crossover operators.

Euclidean Distance to the Ideal Solution The ED indicator is useful to identify which is the solution that has the best trade-off among the optimized objectives, that is usually preferred by decision makers. Table 8 presents the fitness and the ED values of the solution that is closest to the ideal solution by experiment for each SPL. The cells highlighted in gray in each column of the table refer to the three experiments that found the solutions with the lowest ED for each SPL. The lower the ED value, the better the result. It is important to note that the best experiments for AGM and MM are the same, in this order: FdC+MC, FdC+CC and All. For BET, the FdC+CC, CC+MC and All experiments showed the best results. It is important to emphasize that best performing experiments for this quality indicators are those that achieved the best FM values for the three SPLs.

In addition, it is possible to notice that the ED values achieved by the best performing experiments diverge with greater expressiveness in relation to the ED values of the solutions found by the baseline experiment (B) for the PLAs MM and BET. For MM,

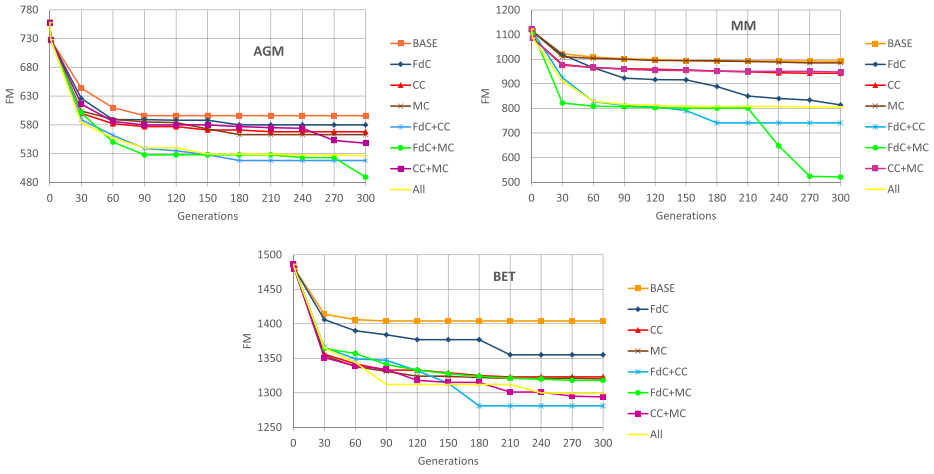

Fig. 7 FM values over generations (the lower value, the better result)

Table 8 Fitness Values (ACLASS, COE, FM) and Euclidean Distance (ED) of the solutions with the lowest ED achieved by experiment

Experiment	AGM		MM		BET	
	Fitness	ED	Fitness	ED	Fitness	ED
Original PLA	(30, 26, 758)	–	(14, 28, 1122)	–	(122, 100, 1486)	–
B	(10, 32, 596)	596.94	(10, 35, 992)	992.66	(128, 97, 1470)	1478.74
FdC	(10, 30, 580)	580.86	(12, 36, 814)	814.88	(129, 95, 1401)	1410.13
CC	(13, 30, 568)	568.94	(10, 33, 943)	943.63	(159, 88, 1325)	1337.40
MC	(13, 30, 563)	563.94	(10, 37, 985)	985.74	(136, 95, 1325)	1335.34
FdC+CC	(14, 28, 518)	518.94	(10, 30, 741)	741.67	(152, 89, 1281)	1293.00
FdC+MC	(12, 27, 489)	489.89	(13, 26, 521)	521.81	(136, 95, 1318)	1328.39
CC+MC	(12, 30, 548)	548.95	(10, 34, 948)	948.66	(166, 85, 1300)	1313.30
All	(13, 27, 526)	526.85	(9, 31, 806)	806.64	(130, 94, 1307)	1316.80

whose original design has lower feature modularization, there was a better improvement in the FM value of the solution with the best trade-off generated by FdC+MC (from 1122 to 521, around 54% of decreasing), when compared to the best trade-off AGM solution (from 758 to 489, that is, around 35% of improvement). For BET, the solution with the best trade-off among the objectives achieved a lower rate of improvement for FM, equivalent to 14% (from 1486 to 1281). Like AGM, the original design of BET also presents high feature modularization. On the other hand, the value of ACLASS (class coupling) was severely compromised in the referred solution. From these observations, we infer that there is a greater range of performance to the crossover operators (especially *Feature-driven Crossover*) for PLAs that originally have worse feature modularization, since there are more features to be modularized.

Since the *Complementary Crossover* operator achieved better results for BET, it is possible to infer that this operator is able to effectively complement more properties of the parents, as BET is the biggest design of our study and has a greater diversity of architectural elements. For MM, it is possible to deduce that *Feature-driven Crossover* has a greater impact on the FM values, as expected, since this operator is concerned with feature modularization. It is important to highlight that the combination of operators showed even better results when compared to the separate application of each operator.

Figure 8 depicts parallel coordinates graphs, where each objective function is represented by a coordinate and each line represents a solution. Hence, Fig. 8 represents the fitness (relative to FM (Feature Modularization), COE (Cohesion) and ACLASS (Class Coupling)) of each solution of the PF_{known} set, that is the best solutions (all the non-dominated solutions) achieved by a particular configuration (experiment) after the 30 independent runs. We improved the explanation about this figure in the text.

In order to complement the results obtained so far, Fig. 8 presents the graphs of the parallel coordinates referring to the baseline experiments (B) and the experiments with the best trade-off for AGM, MM and BET. In the figures, the values of FM, COE and ACLASS are represented in the coordinates of the graphs as Feature Modularization, Cohesion and Coupling, respectively. These graphs present the solutions of the PF_{known} set, which are the best solutions (all the non-dominated solutions) achieved by a particular experiment after the 30 independent runs. As can be seen, a good part of the FdC+MC solutions achieved an FM value less than the minimum of the same objective in B. Furthermore, the maximum

and minimum values of each objective reveal a significant improvement in the solutions generated for the three PLAs.

For MM, the range of values of FM is considerably smaller in the baseline experiment (B) than in FdC+MC (Fig. 8(c) and (d)). Also, the latter reached a slightly lower FM minimum than the baseline experiment. In addition, it is possible to notice that in the case of the FdC+MC experiment, the values of FM are very close to 950 and 1000. Due to this value discrepancy, relatively bad values are obtained when performing normalization of the hypervolume results, reinforcing the fact that the B boxplot reached a better result than the FdC+MC boxplot (Fig. 6(b)). These results corroborate that FdC+MC obtained the best results in terms of FM, also obtaining lower values of ACLASS. Furthermore, FdC+MC is able to reduce the maximum value of ACLASS when compared with the baseline experiment.

For BET, analyzing Fig. 8(b) and (f), there is clearly presented greater concentration of solutions with FM values between 1300 and 1500 and with ACLASS values between 200 and 300 for FdC+CC, reaching a total of optimized solutions much more comprehensive than the baseline experiment (B). Conversely, it is possible to verify most solutions generated by the baseline experiment have ACLASS values higher than 500. In this way, the search space achieved for BET with the FdC+CC is notoriously broader, achieving a greater diversity of optimized solutions and superior to the solutions obtained by the baseline experiment. This behavior for BET corroborates the fact that, because it has a larger design, the *Complementary Crossover* operator handles better with *Feature-driven Crossover* to complement different characteristics in many ways.

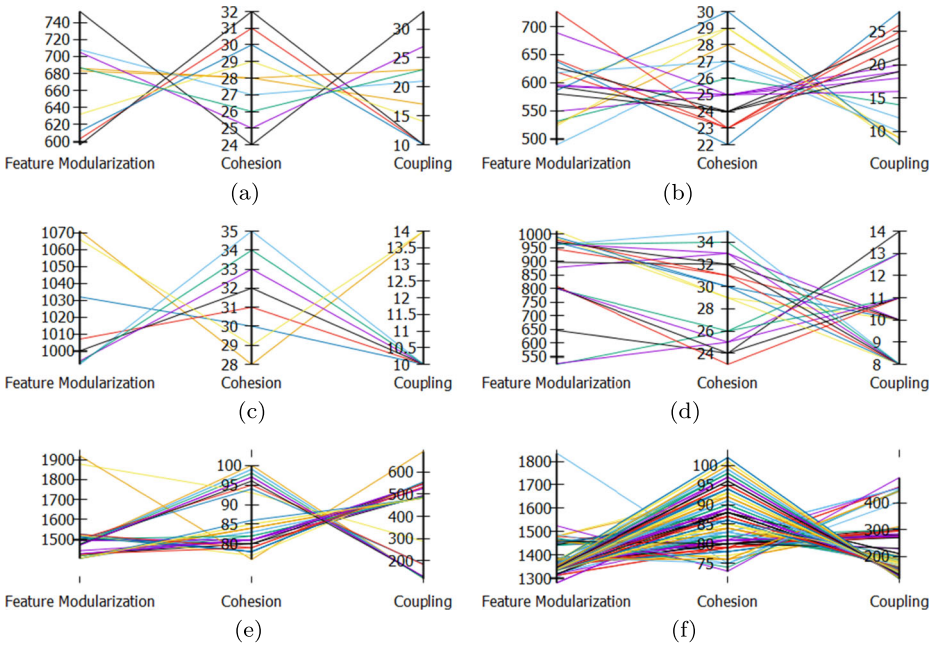

Fig. 8 Parallel Coordinates for B and the experiments that found the solution with the best trade-off among the objectives

Furthermore, when comparing the parallel coordinates of B with the parallel coordinates of FdC+MC and FdC+CC in Fig. 8, it is clear that applying crossover operators contributes to achieve more diversity of solutions as identified in our previous work (Silva et al. 2020).

Overall Analysis of the Quantitative Results Table 9 presents the top three experiments regarding the quality indicators and the FM fitness values in order to summarize the results of the quantitative analysis. The best experiments for AGM and MM are the same considering Coverage, FM fitness values and Euclidean Distance. These experiments are FdC+MC, FdC+CC and All, in this order. There is certain similarity with the results of FM and ED for BET, since FdC+CC, CC+MC and All are the best experiments. Taking into account the Hypervolume indicator, different experiments achieved the highest positions. So, there is not a consensus about which are the best experiment.

An interesting observation is that the *Complementary Crossover* operator has a substantial impact on optimization of the PLA design of BET, as all the best experiments for BET apply this operator, independently of the criterion. Hence, this fact corroborates our inference that such an operator is able to effectively complement the best architectural properties of the parents, especially for BET that is the biggest design of our study. On the other hand, *Feature-driven Crossover* was very beneficial for the optimization of the original design of AGM and MM, since it is applied by almost all the best experiments for both SPLs.

Independently of these observations, we can state that the joint application of at least two of the proposed operators is more beneficial for the optimization of PLA design than the other experiments. This is clear in Fig. 9, which synthesizes the number of the best results achieved by each experiment taking into account the top-3 lists presented in Table 9.

6.2 Qualitative Analysis Regarding Feature Modularization

Aiming at supporting the analysis to answer RQ1.2, in this section we present excerpts of solutions obtained for AGM and MM in order to illustrate the differences regarding feature modularization. For this analysis, we used the solutions with the best trade-off among the optimized objectives. When there is not a preference by a certain objective, the solution closest to the ideal solution is usually preferred by decision makers (Zeleny and Cochrane 1973). As we did not know the user preferences, we choose to use the solution with the

Table 9 Best experiments regarding the quantitative criteria (quality indicators and FM fitness values)

PLA	Quantitative criteria			
	Hypervolume	Coverage	FM fitness	Euclidean distance
AGM	1. FdC	1. FdC+MC	1. FdC+MC	1. FdC+MC
	2. FdC+CC	2. FdC+CC	2. FdC+CC	2. FdC+CC
	3. MC	3. All	3. All	3. All
MM	1. MC	1. FdC+MC	1. FdC+MC	1. FdC+MC
	2. B	2. FdC+CC	2. FdC+CC	2. FdC+CC
	3. CC	3. All	3. All	3. All
BET	1. CC	1. CC+MC	1. FdC+CC	1. FdC+CC
	2. FdC+CC	2. All	2. CC+MC	2. CC+MC
	3. CC+MC	3. CC	3. All	3. All

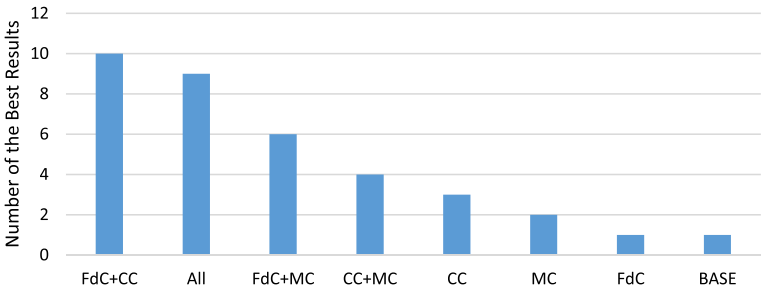

Fig. 9 Number of the best results by quality indicator

lowest EDs (the best trade-off). It is important to highlight that the solutions with the lowest ED are coincidentally those that have the best FM fitness values, as can be seen in Table 8. Therefore, it is possible to analyze the solutions with the best trade-off has also the better feature modularization and which parts of the solutions positively influenced to this behavior.

For AGM, in general, the solution found by the baseline experiment is similar to the solutions of other experiments in terms of feature modularization, with only a few minor differences. However, when comparing the original design with the other experiments, it is easy to see the differences given the new packages created during the optimization to modularize certain features as well as the condensed pre-existing packages, resulting in a reduction in the number of features interlaced with others.

Analyzing Fig. 10(b), it is clear that *Package112450Ctrl* was created to modularize the *ranking* feature in the solution obtained by FdC+MC. In the original design of AGM (Fig. 10(a)), this feature is highly interlaced with the *save* and *play* features. In the solution generated by All (Fig. 10(c)), *ranking* is interlaced with *save*. Thus, for this specific situation, the solution of FdC+MC overcomes the others, since the architectural elements of the package are associated exclusively with *ranking*.

In addition, FdC+CC is the single experiment that improved the modularization of the *configuration* feature, achieving the excerpt of design presented in Fig. 11. It is possible to observe that *configuration* is interlaced with having only one method which is assigned to the *logging* feature. On the other hand, the solution of FdC+CC is the only design in which the *GameBoardCtrl* package is not assigned to several features. In the original design, this is a large package, which contains many classes and interfaces assigned to multiple features.

The solutions found for MM have more significant differences regarding feature modularization in comparison to both the original design and the solution achieved by the baseline experiment, as expected due to the feature modularization degree of the original design. For instance, the *mM* feature has been successfully modularized in the package *UserMgr* for solutions obtained using crossover operators (solutions of FdC+CC, FdC+MC and All), as the elements of this package solely realize *mM*. In the original design, the referred package also realizes part of the *albumManagement* feature. In the baseline solution (B), the interface assigned to *mM* has higher feature interlacing, as it also realizes the *video* and *copyMedia* features.

For the sake of illustration, Fig. 12 depicts excerpts of the obtained solutions that realizes the *favourites* feature. This feature is clearly interlaced with other ones in the original design and in the design obtained by the baseline experiment, as can be seen in Fig. 12(a) and (b), respectively. On the other hand, the positive effect of the crossover operators on the PLA

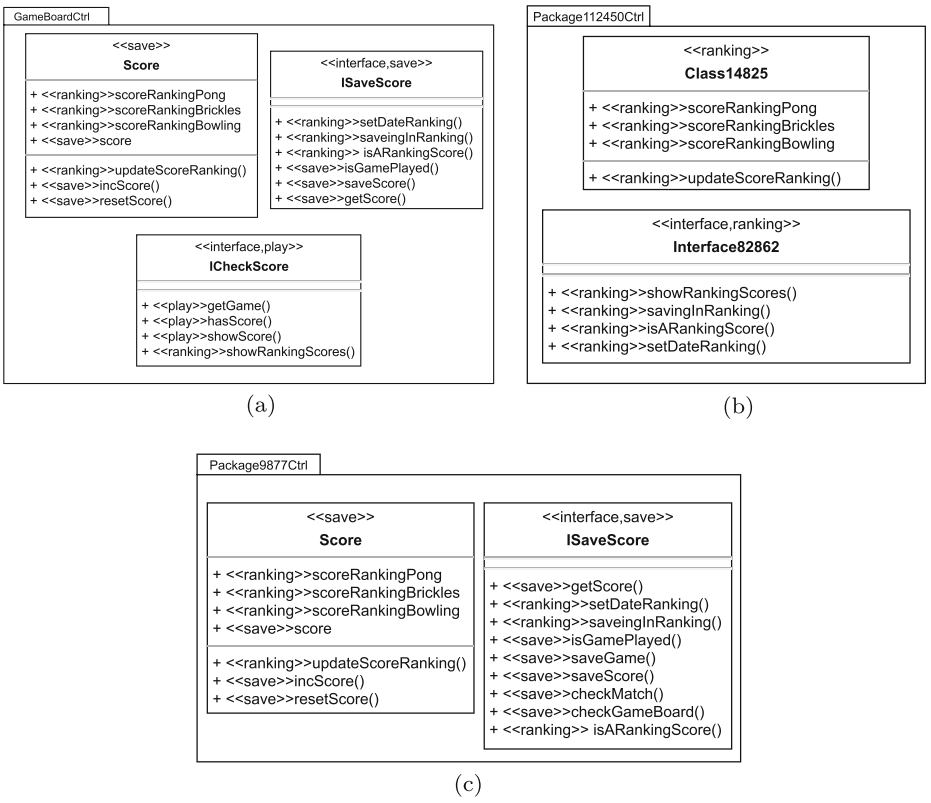

Fig. 10 Modularization of the *ranking* feature for AGM

design optimization in terms of feature modularization can be seen in Fig. 12(c), (d) and (e) representing solutions obtained using crossover operator, where *favourites* is realized by architectural elements assigned exclusively for this feature. However, we can observe in the examples presented in Fig. 12 that some improvements can be done in the obtained solutions. We can illustrate this statement using the *showFavouriteMedias* operation which is inside *IManageFavouriteMedia* interface in Fig. 12(a), (b) and (d) but is not present in the excerpts of the solutions generated by FdC+CC and All (Fig. 12(c) and (e)), what means that this operation assigned to *favourites* is realized by another element increasing the diffusion of this feature.

Another feature that was better modularized using the crossover operators is *sMSTransfer*. The solution obtained by FdC+MC has the highest degree of modularization for this feature. We can observe in Fig. 13 that the excerpt of the design achieved by FdC+MC (Fig. 13(c)) deals exclusively with *sMSTransfer*, while in the design of B (Fig. 13(a)) *sMSTransfer* is interlaced with *photo* in the *ISendMedia* interface. The interfaces presented for FdC+CC and All realize a single feature (*sMSTransfer*), however, there are operations assigned to this feature in other architectural elements, increasing the feature diffusion.

These results MM suggest that the solution with best the trade-off of FdC+MC presents a better feature modularization when compared to both the original design and the solutions

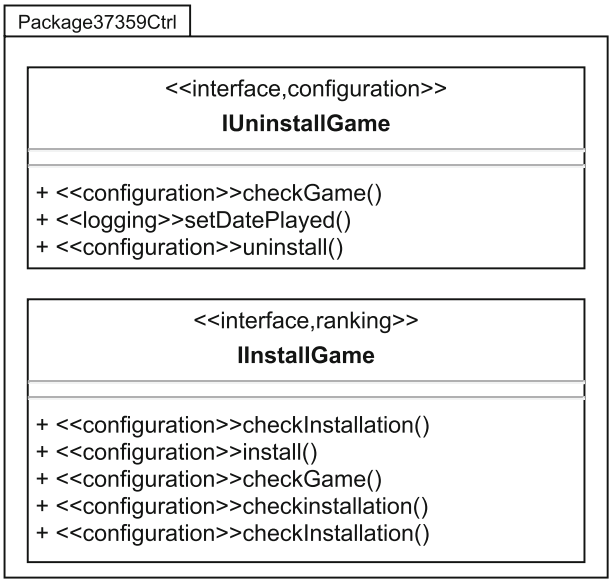

Fig. 11 Modularization of the *configuration* feature in the solution of FdC+CC for AGM

obtained by the other experiments. The solutions generated by FdC+CC and All are similar regarding feature modularization, being superior to the original design and the design obtained by the baseline experiment.

Last but not least, the excerpts of design presented in this section corroborate that the application of crossover operators leads to populations with more diversity, as can be noticed when analysing the different designs achieved by the experiments that apply the proposed crossover operators.

6.3 Qualitative Analysis by Experts

This section presents the results obtained in the evaluation with experts. In what follows, we discuss the results of each question answered by the participants in the order presented in Section 5.3.2.

Relational Cohesion Regarding the baseline solutions, three participants judged that the solution AGM-1 obtained by Exp-B has satisfactory relational cohesion. On the other hand, two participants consider that it has low cohesion as pointed by P4: "*It has low relational cohesion since most classes of some packages are not related*". P1 is more specific when mentioning that "*the classes of Package91599Ctrl and Package90457Ctrl are not so related and in some cases the class that implements an interface is isolated in another package, e.g., GameBoardGUI).*" The baseline solution generated to MM is highly cohesive, according to four experts. Only P2 and P3 suggested improvements, e.g., "*The copyMedia method should not be in IAddMediaAlbum and the media class contains methods that should be in other classes (P3).*"

The experts disagreed about the solutions generated by FdC+CC for AGM and MM (AGM-2 and MM-2). For AGM, two participants (P2 and P3) stated that the cohesion of these solutions is similar to their respective solution obtained by B, whereas P1, P4 and

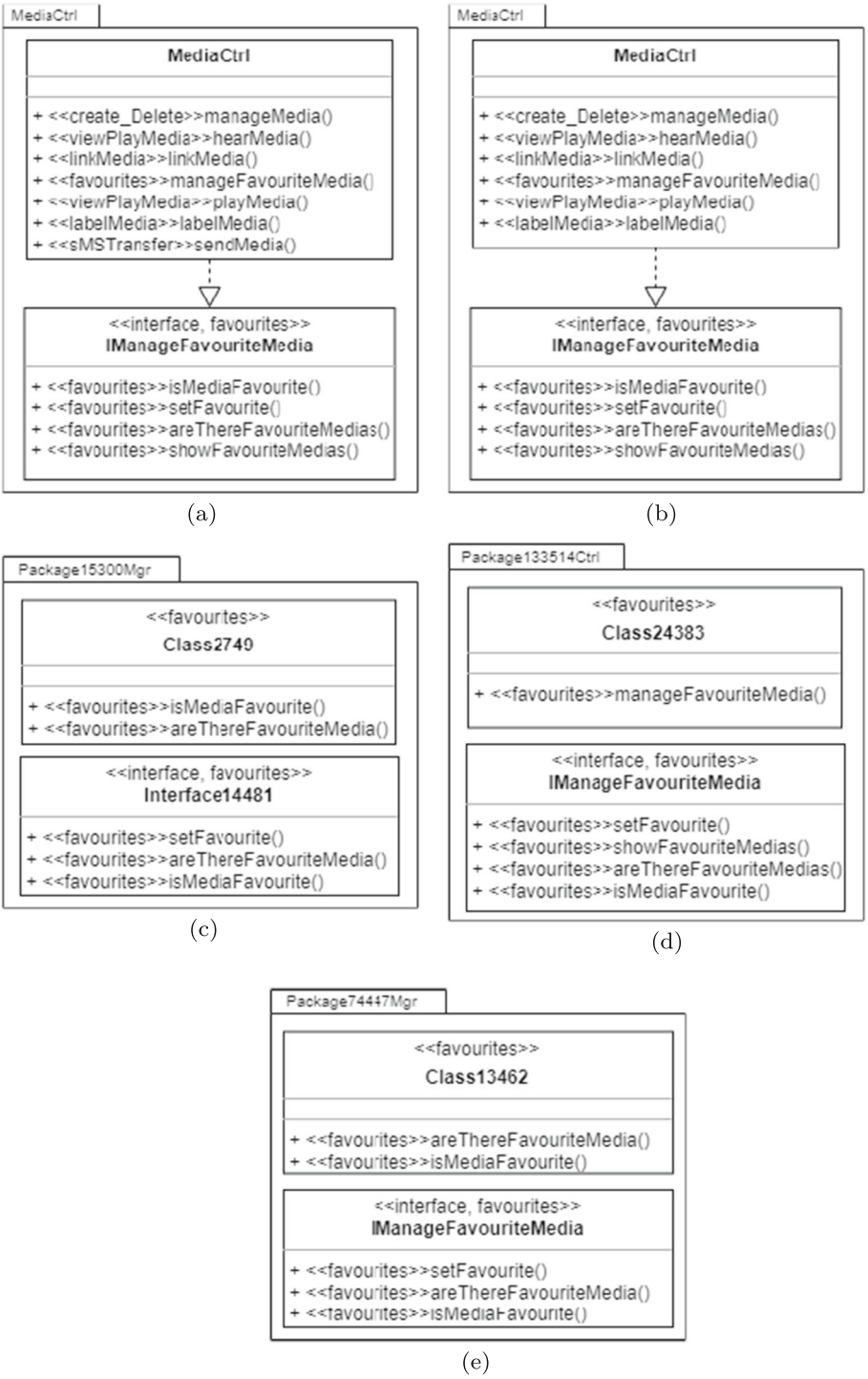

Fig. 12 Modularization of the *favourites* feature in solutions obtained for MM

Fig. 13 Modularization of the *sMSTransfer* features in solutions obtained for MM

P5 judge that the relational cohesion is adequate. Two of them highlighted the effect of the *Feature-driven Crossover* operator. One of them mentioned: "*This solution has few packages, hence higher relational cohesion. However, the modularization of the movement feature negatively impacted the relational cohesion of some elements.*" For MM, P1 stated that this solution is similar to the previous one, P4 stated that the cohesion of MM-2 is similar to MM-1 and the other participants stated that the solution has adequate relational cohesion.

Regarding the solutions achieved by FdC+MC, three participants judged that the solution AGM-3 has satisfactory relational cohesion and four participants stated the same for the solution MM-3. P4 stated that AGM-3 is more cohesive than the AGM-1 and AGM-2 since it has lower number of packages, while P1 judged that MM-3 is slightly more cohesive than MM-1 and MM-2. On the other hand, P1 stated that MM-3 is the lowest cohesive

solution: "*The relational cohesion in this solution is worse than the previous solutions because several packages have classes that are not connected with each other.*"

The solutions obtained by All (AGM-4 and MM-4) were well evaluated with respect to the relational cohesion. AGM-4 and MM-4 were judged as highly cohesive by P1, P2, P4 and P5, and P1 and P5, respectively. P3 stated that AGM-4 has adequate cohesion, while P2, P3 and P4 said that MM-4 has satisfactory cohesion. Despite these evaluations, some experts pointed out that there are improvement opportunities, such as the statement of P5: "*In my opinion, this solution has the best relational cohesion, but package11967Ctrl could be split in two packages to modularize the play and movement features.*"

As can be observed, for relational cohesion, All overcame the other experiments for both AGM and MM, since the solutions were considered highly cohesive in 6 out of 10 evaluations, in accordance with the quantitative results (Table 9). Despite this, the solutions achieved by the other experiments has overall satisfactory relational cohesion.

Coupling Most experts answered the Q2 question regarding class coupling by comparing the four PLA design alternatives presented by SPL. For AGM, two experts (P2 and P5) judged that the solution generated by All (AGM-4) is the best. For example, P5 mentioned: "*After analyzing the four solutions, I judged that all solutions have satisfactory class coupling. However, I think that AGM-4 has the best coupling.*" It is important to recall that as lower the coupling, the better. P3 and P4 judged that the solutions of FdC+CC and FdC+MC achieved the best solutions with respect to coupling. Finally, P1 stated that the solution achieved by B is the best one. Hence, for AGM there is a tie among All, FdC+MC and FdC+CC. These three experiments are the same that have the best performance for AGM in the quantitative analysis (Table 9).

For MM, the solutions generated by the baseline experiment (B) and the experiment that applies all the proposed crossover operators (All) received two votes. In addition, P1 considered that the four PLA design alternatives have satisfactory class coupling. This participant stated: "*MediaMgr and Media are highly coupled in the four solutions. The remaining classes have adequate coupling in all solutions.*" P2, who preferred the solution of B (MM-1) also mentioned the *MediaMgr* package as a highly coupled element in MM-4 (generated by All): "*This design has high coupling. The MediaMgr class has relationships with classes of eight different packages.*" So, for MM, there is also a tie between experiments, but in this case the best performing experiments were B and All.

Interface Size With respect to the Q3 question regarding the interface size, the overall evaluation is that the solutions achieved for AGM and MM have interfaces with adequate size, without responsibility overload according to the expert opinions. However, some interfaces could be segregated and other ones could be grouped.

P1 and P5 suggested segregating the *IGameBoardMgt* and *IGameMgt* interfaces for the four solutions (AGM-1, AGM-2, AGM-3 and AGM-4) since they are assigned to more than one feature. Their suggestion consists of modularizing each feature in a different interface. P1 and P5 also argued that some specific interfaces could have fewer operations in the solution MM-1, e.g., "*Most interfaces have adequate size with few feature interlacing. However, some interfaces, such as IManageMedia and IMediaMgt, could be split in two or more interfaces.*" On the other hand, P2 highlighted the existence of very small interfaces in AGM-4: "*The interfaces are so small leading to a design with several interfaces that have few responsibilities.*" P2 wrote a similar statement for the solutions MM-2 and MM-3. So, the All, FdC+CC and FdC+MC experiments generated small interfaces in some solutions according to the experts' point of view.

Overall, none experiment has overcome each other when considering the interface size.

Feature Modularization The subject of Q4 is the expert opinions about the feature modularization of the evaluated solutions. The baseline solutions were the worst evaluated ones for both AGM and MM. The most common problems mentioned by participants are feature diffusion and feature interlacing over architectural elements, as pointed by P1 with respect to MM-1: "*Many features are diffused over different architectural elements. In addition, there is high feature interlacing in some elements, e.g., Media, IMediaMgt and MediaCtrl). I think the feature modularization of this solution might be significantly improved.*"

The experts agreed that the solutions generated by FdC+CC for AGM and MM (AGM-2 and MM-2) have better feature modularization than the baseline solutions. They argued that the features are less diffused than the baseline solutions and that there is lower feature interlacing. They also mentioned that some elements could be changed to improve feature modularization, as P5 suggested of splitting *IManageMedia* in MM-2 to modularize different features in different interfaces.

The solutions generated by FdC+MC (AGM-3 and MM-3) and All (AGM-4 and MM-4) were pretty well evaluated by the participants. Most experts judged that these solutions have high feature modularization, as highlighted by P2 with respect to AGM-3: "*The variation points are well distributed as well as the features are well modularized in packages with well-defined responsibilities.*" and by P5 with respect to AGM-4: "*This solution has the best feature modularization in both classes and packages.*" Despite the positive evaluation, P2 and P4 presented few suggestions to improve the feature modularization of some new packages and interfaces created during the optimization process by the *Feature-driven Mutation Operator*, e.g., P2 stated that despite the high feature modularization of MM-3, three new packages are too small and deserve further attention. Another negative point raised by P1 about AGM-3 is that: "*There is low feature interlacing as there are more (small) architectural elements. However, the features are diffused over several elements, leading to a bad feature modularization, in my opinion.*"

One participant (P3) considered that all solutions have the satisfactory feature modularization degree for AGM and MM, dealing with one or two features. However, he mentioned: "*Some few elements are assigned to up 4 features, for which I suggest re-designing such elements. Thus, the overall feature modularization of all solutions is acceptable*". He also suggested some improvements in managers and their respective interfaces (elements with suffix Mgr and Mgt) in the solutions obtained for MM.

In summary, All and FdC+MC overcame the experiments B and FdC+CC for both SPLs regarding feature modularization, since their solutions were considered with high feature modularization by most participants. All and FdC+MC are also two of the best performing experiments in the quantitative analysis for AGM and MM.

Which Solution has the Best Feature Modularization? For AGM, the solution obtained by FdC+CC was chosen by the participant P1. The solutions generated by FdC+MC and All were chosen twice. For MM, the solutions of FdC+MC and All received two votes, while the participant P5 preferred the solutions obtained by B and FdC+CC. For example, P2 said: "*AGM3 is better modularized than the other alternative solutions due to the distribution of features over architectural elements*". P1 presented an improvement suggestion: "*I think AGM-2 is the best, taking into account the analyzed criteria. However, it needs some improvements, such as splitting GameBoardCtrl in two or more different packages.*" This statement reinforces our analysis, presented in Section 6.2, that the *GameBoardCtrl*

package of the solution generated by FdC+CC is better than the original version of this package, but it is still realizing more than one feature.

It is interesting to notice that P2 evaluated the solution obtained by FdC+MC for MM as the best solution regarding feature modularization. He took into account the novel packages *Package132839Mgr*, *Package133087Mgr*, *Package132722Mgr* created during the optimization process to modularize the *video*, *copyMedia* and *photo* features, respectively. This is the solution that has the lowest Euclidean distance to the ideal solution, as can be observed in Table 8. The referred solution has also the lowest fitness values for feature modularization (FM objective function) and relational cohesion (COE objective function). This fact points that even without knowing the fitness values and the results of the quality indicators, two experts chose the best solution according to quantitative results.

Another interesting observation is that in spite of P3 has chosen MM-4 obtained by All for answering Q5, he mentioned, in the previous question (Q4), that MM-4 and MM-2 (FdC+CC) are equivalent in terms of feature modularization and he untied both solutions considering the class coupling. Both solutions have slightly better quantitative results than the other three solutions.

Another important finding is related to the statement of P1 that there is high feature interlacing in some elements of the solution MM-1 obtained by B, e.g., *IMediaMgt* interface of *MediaMgr* package. When comparing this interface in MM-1 and in MM-4—the solution chosen by P1 as the best one regarding feature modularization—it is noticeable that *IMediaMgt* of MM-1 has methods assigned to different features, such as *sMSTransfer*, *favourites*, *viewPlayMedia* and *create_Delete*, as can be seen in Fig. 14(a). On the other hand, the *IMediaMgt* interface of the solution obtained by All (MM-4) deals with a single feature, namely *viewPlayMedia* (depicted in Fig. 14(b)). Furthermore, the feature interlacing of *MediaMgr* is lower in MM-4 than in MM-1, dealing with the features *music* and *viewPlayMedia*.

In summary, All and FdC+MC received four votes for the question Q5, what means that the experiments that apply all the proposed crossover operators (All) and the experiment that apply the *Feature-driven Crossover* and *Modular Crossover* operator (FdC+MC) have the best performance in terms of the solution with best feature modularization. Once again, these experiments are also the best ones in the quantitative analysis for AGM and MM.

Architectural Properties to be Evaluated in PLA Design In the Q5 question, we were interested in knowing what are the architectural properties that the experts judge to be essential to be evaluated in a PLA design. This kind of information is useful to validate whether the objective functions used during the optimization process are appropriate in the point of view of experts. More importantly, this is a rich opportunity to discover other important properties to be included in the evaluation model of MOA4PLA.

The five participants mentioned that feature modularization is the main property to be evaluated in a PLA design. P1 emphasized that "*the PLA design needs to have excellent feature modularization in order to enable the reuse of SPL assets and ease the variability management.*" Furthermore, P3 explicitly mentioned lack of feature-based cohesion when argued that we need to evaluate the number of features of each architectural element.

P2, P3 and P4 also cited architectural properties that are the goal of the objective functions of MOA4PLA, such as coupling, cohesion and interface size, as highlighted by P4: "*The main properties to be evaluated are (in this order) 1. feature modularization, 2. coupling, 3. cohesion.*"

In addition, P2 and P3 mentioned the number of class responsibilities and the class size, respectively, which are not measured and optimized by the objective functions of MOA4PLA.

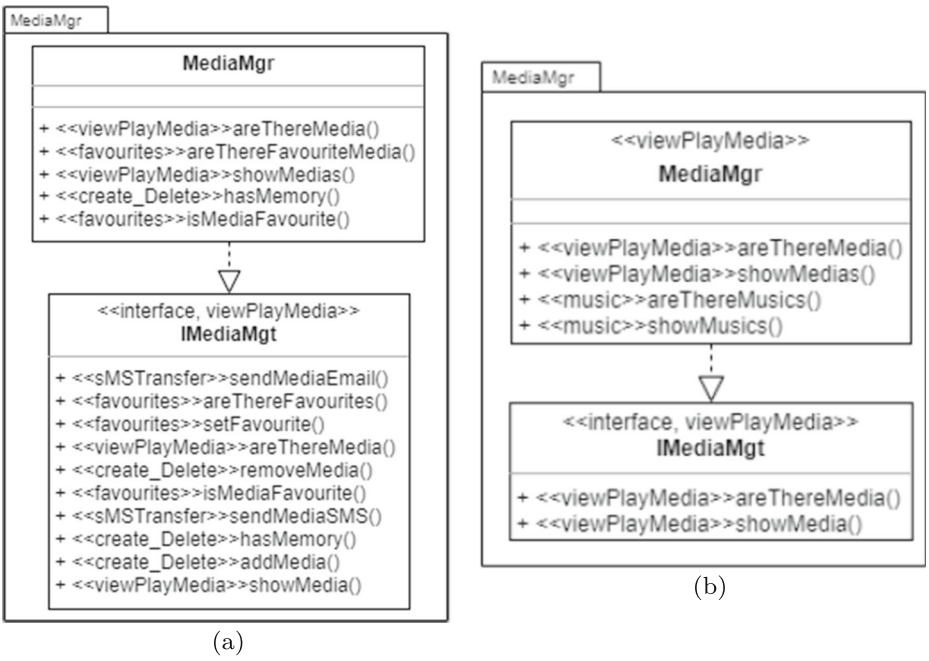

Fig. 14 Comparison of feature modularization of excerpts of PLA design solutions obtained by B and All

Overall Analysis of the Qualitative Results The results of the qualitative analysis with experts indicate a greater acceptance by the participants for the All and FdC+MC experiments, as can be seen in Table 10, which presents the experiments with better performance by criterion evaluated in the study.

Regarding the feature modularization, the All and FdC+MC were equivalent for AGM and MM. Regarding relational cohesion, the solutions obtained by the experiment that applied all crossover operators (All) were more accepted for both PLAs. In relation to the class coupling, there were divergent opinions, however, the solutions achieved by All were well evaluated by experts for both AGM and MM. Therefore, it is possible to verify that the joint application of the proposed crossover operators generated solutions positively evaluated by the experts. As FdC+MC also obtained acceptance equivalent to All regarding the feature modularization, and both operators are applied in All, it is noted that *Feature-driven Crossover* and *Modular Crossover* generate the most appropriate solutions from their point

Table 10 Experiments with better performance in the qualitative evaluation according to the experts

PLA	Relational cohesion	Class coupling	Interface size	Feature modularization
AGM	All	All FdC+CC FdC+MC	–	All FdC+MC
MM	All	All B	–	All FdC+MC

of view. It is interesting to note that the solutions best accepted by the experts had the best results in quantitative analysis (mainly FdC+MC), corroborating this finding.

6.4 Answering the Research Questions

In this section we answer the posed research questions taking into account the results presented in the previous subsections.

> ***RQ1 - What is the benefit of using crossover operators in the optimization of PLA design in comparison to the state of the art?***

The results obtained in the empirical study showed that the experiments that apply crossover operators are almost always better than the baseline experiment. Given the empirical results, we can state that the benefit of using crossover operators in the PLA design optimization are threefold: (i) combine/merge parts already optimized in different individuals; (ii) lead to more diversity in the population of potential PLA designs; and (iii) generate solutions with better feature modularization. These benefits are discussed in more details in the next sub-RQs.

> ***RQ1.1 - What is the performance of the proposed crossover operators for PLA design optimization in terms of quality indicators?***

The overall analysis of the quantitative results showed that the FdC+CC, All and FdC+MC experiments were the best. Despite the divergence between the hypervolume results and the results of the other quality indicators, FdC+MC obtained the best result for AGM and MM and, in general, we can state that the *Feature-driven Crossover* operator was very beneficial for both SPLs. Taking into account the results achieved for BET, the best experiment was CC+MC and the *Complementary Crossover* operator was significantly beneficial for this SPL. However, as the best performing experiments were always those that combine more than one crossover operator, we can state that the joint application of at least two of the proposed operators is more beneficial for the optimization of PLA design than the other experiments. Taking into account the different purpose of each crossover operator, the better results achieved by FdC+CC, All and FdC+MC evidenced that the proposed crossover operators complement each other, optimizing the solutions more broadly in relation to the different objectives.

> ***RQ1.2 - What is the impact of the crossover operators on feature modularization of the PLA design solutions?***

The experiments that apply the *Feature-driven Crossover* were the best performing ones in the quantitative analysis, what points out that this operator is able to optimize the PLA design since one of the objective functions is related to feature modularization (FM). More importantly, the solutions obtained using crossover operators have better feature modularization than the original design and the baseline solution, including the experiments that

do not apply the *Feature-driven Crossover*, as can be noticed in Table 8 and in Fig. 8. For instance, CC+MC achieved a lower value of FM than the baseline solution. As aforementioned, for AGM and BET the improvement impact on feature modularization was lower than for MM, because the original design of the two former SPLs have better feature modularization than MM. In addition, it is possible to verify that the experiments that applied the *Feature-driven Crossover* achieved greater gain in feature modularization for MM. Hence, a worse modularized design has a greater search space to be explored by *Feature-driven Crossover*. Furthermore, when considering the ED quality indicator and the FM fitness values (Table 9), the best performing experiments are those that apply more than one crossover operator, emphasizing that the combination of *Feature-driven Crossover* with other crossovers is more effective to improve the feature modularization of the PLA design under optimization.

RQ1.3 - Which crossover operator generates better solutions according to the experts?

The results of the qualitative analysis showed that the *Feature-driven Crossover* and *Modular Crossover* operators generate the most appropriate solutions from the point of view of experts who participate in our study. The fact that the experts preferred the solutions generated by experiments that apply more than one crossover operator (All and FdC+MC) corroborates the evidence that the proposed crossover operators complement each other and that they are beneficial for search-based PLA design. The best accepted solutions by the experts achieved the best results in the quantitative analysis (mainly FdC+MC), corroborating these findings. In addition, the experts' evaluation reveals that the obtained solutions need some polishing before being adopted as the PLA of the SPL under development, as we also detected and pointed in Section 6.2. Indeed, when using SBSE techniques, it is expected the automatic generation of near-optimal solutions, which needs some adaptations before using in practice.

7 Threats to Validity

In this section, we discuss the threats to validity related to our quantitative and qualitative empirical studies, based on the taxonomy of Wohlin et al. (2000). We also describe how we mitigate possible threats.

Internal Validity An internal threat is the fact that the search-based algorithm used in the experiments is non-deterministic. In order to mitigate this threat, each experimental configuration was performed 30 times for the three subject PLAs. Another internal threat is the size and diversity of the subject PLA designs. Regarding the diversity of PLAs, it is possible to affirm that the three PLAs are distinguished in relation to their sizes, in which BET is much larger than the others; and domains. This allowed us to analysis the results from different perspectives and aspects. In this sense, we believe this treat is reduced. The last treat is related to the selection of SPL experts to participate in the qualitative experiment. Despite not conducting an extensive search for experts, we believe that the five ones that participate in our experiment are representative to allow us to reach relevant results.

External Validity The two main threats related to the generalization of the results are regarding the subject PLA designs and the opinion of experts. Despite the AGM and MM being academic PLAs, their development took into account real scenarios. Additionally, BET was developed for practical purposes, which makes it very close to an industrial system. We can also highlight that finding industrial projects with the necessary details to carry out the experiments is not a trivial task. Taking into account the opinion of experts, despite only five participants, they all have at least a moderate experience in UML design and SPL engineering. And, despite being students, three of them are/have been also in industry, which brings even more credibility to their opinion. Overall, excluding organizational and operational aspects that vary from company to company, our results can be generalized.

Construct Validity To avoid threats related to how the solutions were obtained, we used objective functions based on software metrics of coupling, cohesion, and modularity, which are widely used for designing architectures. Also, these metrics were used in several previous studies. In addition to the metrics, the crossover operators were designed based in accordance with existing theory for creating software architectures. To mitigate threats related to parameters settings, we adopted crossover and mutation rates experimentally defined in our previous work.

Conclusion Validity Finally, we alleviated all threats to conclusion validity by using quality indicators widely adopted in the field of multi-objective optimization. For the quantitative analysis, statistical and effect size tests were applied to avoid any subjective interpretation of the results. Also, the qualitative analysis as conducted in a way to prevent any bias that could impact the opinion of the participants.

8 Concluding Remarks

In this paper, we proposed three crossover operators whose goal is intensifying the search-based PLA design. The proposed operators are: *Feature-driven Crossover*, *Complementary Crossover* and *Modular Crossover*. *Feature-driven Crossover* aims at improving the feature modularization of the PLA design. The purpose of *Complementary Crossover* is extracting better characteristics of different parents and merge in the offspring. The goal of *Modular Crossover* is preserving structural allocations of complete or partial modules from a parent to a child. In addition, the proposed operators are able to generate complete and consistent solutions, improving the state of the art.

We performed an empirical study, where we compared the results obtained using the proposed crossover operators against the state of the art. This empirical study also included a qualitative analysis of the generated solutions by experts in order to identify which are the best solutions in their point of view and to detect improvement needs. The empirical results showed that there are significant differences among the use of only mutation (state of the art), and mutation with crossover and that the crossover operators contributed to generate solutions with better feature modularization. Results also showed that the proposed operators complement each other, as the experiment that combines at least two of the proposed operators achieved better results.

In a future work, we intend to investigate: (i) the evidence that the *Complementary Crossover* is more beneficial for large PLA designs, (ii) the high impact of the *Feature-driven Crossover* on other original designs that have poor feature modularization, (iii)

conducting an exploratory study with interviews to obtain more insights from experts on how to improve PLA optimization.

Acknowledgements The work is supported by CNPq Grants No. 428994/2018-0 and No. 408356/2018-9, CAPES Grant No. 448977/2019-01, and FAPERJ PDR-10 program under Grant No. 202073/2020.

Declarations

Conflict of Interest The authors declare that they have no conflict of interest.

References

Bass L, Clements P, Kazman R (2003) Software architecture in practice. Addison-Wesley Professional

Choma Neto J, Gaieski T, Amaral AM, Colanzi TE (2018) Quanti-qualitative analysis of a memetic algorithm to optimize product line architecture design. In: 2018 IEEE 30th international conference on tools with artificial intelligence (ICTAI), pp 498–505. https://doi.org/10.1109/ICTAI.2018.00083

Choma Neto J, da Silva CH, Colanzi TE, Amaral AM (2019) Are memetic algorithms profitable to search-based PLA design? IET Softw 13(6):587–599. https://doi.org/10.1049/iet-sen.2018.5318

Colanzi TE, Vergilio SR (2016) A feature-driven crossover operator for multi-objective and evolutionary optimization of product line architectures. J Syst Softw 121:126–143. https://doi.org/10.1016/j.jss.2016.02.026

Colanzi TE, Vergilio SR, Gimenes I, Oizumi WN (2014) A search-based approach for software product line design. In: Proceedings of the 18th international software product line conference (SPLC'14), vol 1, pp 237–241. https://doi.org/10.1145/2648511.2648537

Colanzi TE, Assunção WK, Vergilio SR, Farah PR, Guizzo G (2020) The symposium on search-based software engineering: past, present and future. Inf Softw Technol 127:106372. https://doi.org/10.1016/j.infsof.2020.106372

Contieri AC Jr, Correia GG, Colanzi TE, Gimenes IM, Oliveira EA Jr, Ferrari S, Masiero PC, Garcia A (2011) Extending uml components to develop software product-line architectures: lessons learned. In: Proceeding of the 5th European conference on software architecture (ECSA), pp 130–138. https://doi.org/10.1007/978-3-642-23798-0_13

Czarnecki K (2013) Variability in software: state of the art and future directions. In: International conference on fundamental approaches to software engineering, FASE'13, pp 1–5. https://doi.org/10.1007/978-3-642-37057-1_1

Deb K (2015) Multi-objective evolutionary algorithms. In: Springer handbook of computational intelligence. Springer, pp 995–1015. https://doi.org/10.1007/978-3-662-43505-2_49

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Trans Evol Comput 6(2):182–197. https://doi.org/10.1109/4235.996017

Donegan PM, Masiero PC (2007) Design issues in a component-based software product line. In: SBCARS, pp 3–16

Féderle É L, do Nascimento Ferreira T, Colanzi TE, Vergilio SR (2015) Opla-tool: a support tool for search-based product line architecture design. In: Proceedings of the 19th international software product line conference, pp 370–373. https://doi.org/10.1145/2791060.2791096

Freire WM, Massago M, Zavadski AC, Malachini AM, Amaral M, Colanzi TE (2020) OPLA-tool v2.0: a tool for product line architecture design optimization. In: Proceedings of the 34th Brazilian symposium on software engineering, SBES '20. Association for Computing Machinery, New York, pp 818-823. https://doi.org/10.1145/3422392.3422498

Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning, 1st edn. Addison-Wesley, Longman Publishing Co., Inc.

Gomaa H (2011) Software modeling and design: UML, use cases, patterns, and software architectures, Cambridge University Press, Cambridge. https://doi.org/10.1017/CBO9780511779183

Guizzo G, Colanzi TE, Vergilio SR (2017) Applying design patterns in the search-based optimization of software product line architectures. Softw Syst Model. https://doi.org/10.1007/s10270-017-0614-9

Harman M, Hierons R (2002) A new representation and crossover operator for search-based optimization of software modularization. In: Proceedings of the annual conference on genetic and evolutionary computation (GECCO'2002), pp 1351–1358

Harman M, Tratt L (2007) Pareto optimal search based refactoring at the design level. In: 9th Annual conference on genetic and evolutionary computation. ACM, New York, pp 1106–1113. https://doi.org/10.1145/1276958.1277176
Juristo N, Moreno AM (2013) Basics of software engineering experimentation. Springer Science & Business Media
Kitchenham B, Charters S (2007) Guidelines for performing systematic literature reviews in software engineering. In: Proceedings of the 2nd international workshop on evidential assessment of software technologies (EAST'12)
Knowles JD, Corne DW (2000) Approximating the nondominated front using the pareto archived evolution strategy. Evol Comput 8(2):149–172. https://doi.org/10.1162/106365600568167
Kruskal WH, Wallis WA (1952) Use of ranks in one-criterion variance analysis. J Am Stat Assoc 47(260):583–621
Li M, Yao X (2019) Quality evaluation of solution sets in multiobjective optimisation: a survey. ACM Comput Surv 52:1–38. https://doi.org/10.1145/3300148
Mishra P, Pandey CM, Singh U, Gupta A, Sahu C, Keshri A (2019) Descriptive statistics and normality tests for statistical data. Ann Cardiac Anaesth 22(1):67. https://doi.org/10.4103/aca.ACA_157_18
Nunes C, Kulesza U, Sant'Anna C, Nunes I, Garcia A, Lucena C (2009) Assessment of the design modularity and stability of multi-agent system product lines. J Univ Comput Sci 15(11):2254–2283
Räihä O, Koskimies K, Mäkinen E (2009) Empirical study on the effect of crossover in genetic software architecture synthesis. In: Proceedings of world congress on nature and biologically inspired computing, pp 619–625
Räihä O, Koskimies K, Mäkinen E (2010) Complementary crossover for genetic software architecture synthesis. In: Proceedings of the 10th international conference on intelligent systems design and applications, pp 266–271. https://doi.org/10.1109/ISDA.2010.5687255
SEI: Arcade game maker pedagogical product line (2009) https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=485941
Silva DFD, Okada LF, Colanzi TE, Assunção WKG (2020) Enhancing search-based product line design with crossover operators. In: Genetic and evolutionary computation conference (GECCO '20), pp 1250–1258. https://doi.org/10.1145/3377930.3390215
Simons C, Parmee IC, Gwynllyw R (2010) Interactive, evolutionary search in upstream object-oriented class design. IEEE Trans Softw Eng 36(6):798–816. https://doi.org/10.1109/TSE.2010.34
Surhone LM, Timpledon MT, Marseken SF (2010) Shapiro-Wilk test. VDM Publishing. https://books.google.com.br/books?id=LsG-cQAACAAJ
Van der Linden FJ, Schmid K, Rommes E (2007) Software product lines in action: the best industrial practice in product line engineering. Springer Science & Business Media. https://doi.org/10.1007/978-3-540-71437-8
Vargha A, Delaney HD (2000) A critique and improvement of the cl common language effect size statistics of McGraw and Wong. J Educ Behav Stat 25(2):101–132
Verdecia YD, Colanzi TE, Vergilio SR, Santos MCB (2017) An enhanced evaluation model for search-based product line architecture design. In: Proceedings of the XX Ibero-American conference on software engineering (CIbSE - ICSE 2017), pp 155–168
Wohlin C (2014) Guidelines for snowballing in systematic literature studies and a replication in software engineering. https://doi.org/10.1145/2601248.2601268
Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2000) Experimentation in software engineering: an introduction. Kluwer Academic Publishers USA. https://doi.org/10.1007/978-1-4615-4625-2
Young TJ (2005) Using aspectj to build a software product line for mobile devices. Master's thesis, University of British Columbia
Zeleny M, Cochrane JL (1973) Multiple criteria decision making. University of South Carolina Press
Zitzler E, Thiele L (1998) Multiobjective optimization using evolutionary algorithms—a comparative case study. In: Eiben AE, Bäck T, Schoenauer M, Schwefel HP (eds) Parallel problem solving from nature—PPSN v. Springer, Berlin, pp 292–301
Zitzler E, Laumanns M, Thiele L (2002) SPEA2: improving the strength pareto evolutionary algorithm. In: Evolutionary methods for design, optimization and control with applications to industrial problems (EUROGEN), pp 95–100
Zitzler E, Thiele L, Laumanns M, Fonseca CM, Da Fonseca VG (2003) Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans Evol Comput 7(2):117–132