Empirical Software Engineering (2021) 26: 58
https://doi.org/10.1007/s10664-021-09947-7

Genetic programming for feature model synthesis: a replication study

Andreea Vescan[1] · Adrian Pintea[1] · Lukas Linsbauer[2] · Alexander Egyed[3]

Accepted: 30 December 2020 / Published online: 21 April 2021

Abstract

Software Product Lines (SPLs) make it possible to configure a single system based on features in order to create many different variants and cater to a wide range of customers with varying requirements. This configuration space is often modeled using Feature Models (FMs). However, in practice, the SPL (and consequently the FM) is often created after a set of variants has already been created manually. Automating the task of reverse engineering a feature model that describes a set of variants makes the process of adopting an SPL easier. The genetic programming pipeline is a good fit for feature models and has been shown to produce good reverse engineering results. In this paper, we replicate the results of such an existing approach with a larger set of feature models and investigate the effects of various genetic programming parameters and operators on the results. The design of our replication experiments employs three perspectives: duplicate the exact conditions using various features models, study the interaction of two parameters of the genetic programming approach, and optimize the values for the population and generation parameters and for the mutation and crossover operators. Results reinforce the previously obtained outcome, the original study being confirmed. The relations between the number of features and number of generations, respectively number of features and size of populations were also investigated and best values based on obtained results are provided. The current study also aimed to optimize various parameters of the genetic programming approach, the interpretation of those experiments discovering concrete values.

Keywords Feature models · Replication study · Reverse engineering · Software product lines

Communicated by: Laurence Duchien, Thomas Thüm and Paul Grünbacher

This article belongs to the Topical Collection: *Configurable Systems*

✉ Alexander Egyed
alexander.egyed@jku.at

Extended author information available on the last page of the article.

1 Introduction

This paper aims to replicate the results of an existing approach for reverse engineering of Feature Models (FMs) for Software Product Lines (SPLs) (Linsbauer et al. 2014). SPLs can provide assistance in providing support for systems that are developed for/with multiple customers in mind - systems that are typically customizable reflect different customer requirements. Reverse engineering of feature models in the domain of SPLs provides engineering support for the systems developers. However, creating a feature model for an existing system manually is time-consuming and requires substantial effort. By providing automated support for the reverse engineering of feature models, developers thus benefit from reduced effort.

As specified in Linsbauer et al. (2014), “a crucial step in this reverse engineering effort is obtaining a feature model that denotes all the desired feature combinations”. This feature model contains all the possible and valid combinations of features. There are various studies (Haslinger et al. 2011; Acher et al. 2012; Haslinger et al. 2013; Acher et al. 2013; Linsbauer et al. 2014; She et al. 2014; Lopez-Herrejon et al. 2015; Assunção et al. 2016) that explored this problem, using different approaches from graph based algorithms to search based techniques. Details regarding those approaches are further provided in Section 2.1 of this paper.

One of the solutions for this problem is genetic programming pipeline which is described in Linsbauer et al. (2014), where the authors showed that genetic programming provides a more accurate representation of the feature models, and that this representation can produce better reverse engineering results. We underline that a list of feature sets can be expressed by different feature models, thus the considered feature order influences the obtained results.

The original work by Linsbauer et al. (2014) provides not only automation but also empirical evidence about its usefulness. With this replication study, we aim to confirm or reject their results. We also expand on that work to generalize it further and to reduce the original threats to validity. Replication studies are an indispensable element of the experimental paradigm: replication in different contexts, at different time and under different conditions being needed to produce generalized knowledge. But it also has been seen as an important means of assessing reliability and confidence in empirical findings in previous studies. Thus, a replication is confirmatory (i.e., increasing the confidence in the results of earlier replications) or it yields additional knowledge (Juristo and Vegas 2009) to the software engineering community (i.e., finding out more about the range of conditions under which the results hold).

These results require robust replication to assess their validity and to generalize the findings. Replication is a central part of accumulating reliable evidence (Dyba et al. 2005). For these reasons, this study aims to improved rigor by replicating the original study.

The current study pursues three main research questions: RQ1: Are the original results of the original study confirmed in other samples?

RQ2: Are there differences in using various numbers for $\#generations$ and $\#population$?

RQ3: Do values for crossover and mutations play a role in obtaining better solutions?

We replicated the original study by Linsbauer et al. (2014) to investigate the obtained Genetic Programming (GP) solutions for the reverse engineering problems, to find which are the optimum values for the $\#generations$ (number of generations) and $\#population$ (population number), and also to discover the best experiment parameters, i.e. crossover and mutation of the GP approach.

One reason for choosing this paper for replication was the availability of implementation and data. It was the first study that the authors conducted relating to feature model synthesis that had a replication package available.

We did not consider an extended version (Assunção et al. 2016) of the selected study (Linsbauer et al. 2014) for replication as it takes into account implementation level dependencies for the optimization and uses multi-objective optimization. Replication of such a study would be much more difficult and we would need to control for many more aspects. We wanted to first replicate the initial study (Linsbauer et al. 2014) to get a better understanding of the foundations before moving on to other studies and approaches that build on the fundamental ones.

The replication results confirm the previous obtained results regarding the genetic programming approach used for the automatic reverse engineering of feature models. The obtained results also demonstrate strong relations between number of features and the number of generations and between number of features and the number of population. The replication contributes with new information regarding these relations, providing concrete values for those parameters to run the algorithm with when the number of features increases. The study also analyzes and scrutinizes the best optimized values for several parameters of the genetic programming algorithm.

Thus, the results provide strong empirical support for the relevance and importance of automatic reverse engineering of feature models. Therefore, this study provides a methodological basis for replicating automatic feature models reverse engineering studies in Software Engineering.

The paper is organized as follows: Section 2 briefly describes the notions and concepts related to feature models, the original study, how to do replication in software engineering, and presents the motivation for the replication. Section 3 describes in more details our replication approach, stating the research questions and the steps in our replication design, followed by the analysis of experiments and the used data analysis. Section 4 presents the dataset for the replication, the steps for experiments executions and the obtained results. Section 5 discuss about the threats to validity that can affect the results of our study. The conclusions of our paper and further research directions are outlined in Section 6.

2 Background

The current section briefly depicts the notions and concepts related to feature models, their importance and presents various synthesis approaches. The original study is also discussed in details. A subsection is dedicated to how to conduct replication in Software Engineering, whereas the last part presents the motivation for this replication.

2.1 Feature Model Synthesis

A feature model is a compact representation of all the products of the Software Product Line (SPL) in terms of “features”. The variability of software systems is designed with the help of feature models. A Feature model is composed of a set of boolean constraints, a hierarchical tree representation of parent-child relationships that are established between features, and a constraints set between features. These feature models tell us which combinations of features are allowed and which combinations are prohibited. Unfortunately, companies only use this reverse engineering of an SPLs from their existing product variants when the number of possible configurations of a product is difficult to manage.

Feature models are important for changing the variability of a system because these feature models describe all the possible and valid combinations of features. The features of a system can be mandatory or optional and can be described with the help of some labelled boxes which are connected together, forming a tree structure. In a feature model, each feature has a parent feature except for the root of the tree structure that is always included in a model.

Several types of relationships can be established between a feature that is considered to be a child feature and a parent feature. Whenever a parent feature is selected, mandatory features are also selected unlike the optional features which may or may not be selected whenever a parent feature is selected. In an "exclusive-or" relationship only one feature is selected along with the selected parent. In a "inclusive-or" relationship at least one feature is selected along with the selected parent. Along with these relations between parent and child features, the so called Cross-Tree Constraints (CTCs) helps to create relationships between different branches of the feature model.

A Feature Set is a combination of features. This set is valid if all the constraints imposed by the feature model are respected. A feature set table contains all the valid feature sets of the feature model. A feature set table therefore represents a set of products, each product having a different combination of features (i.e., feature set).

Figure 1 shows the Feature Diagram for the "SmartHome" feature model. As mentioned above, features are described with the help of labelled boxes. The Cross-Tree Constraints set is exposed under the Feature Diagram. The mandatory features are WindowsControl, ManualWindows, HeatingControl, UI, InhomeScreen, and the optional features are AutomatedWindows, SmartHeating, InternetApp, Security, BurglarAlarm, Siren, Bell, Light, Notification, Authorization, Authentication, LAN and DoorLock. An inclusive-or relationship can be observed between Siren, Light, Bell and Notification.

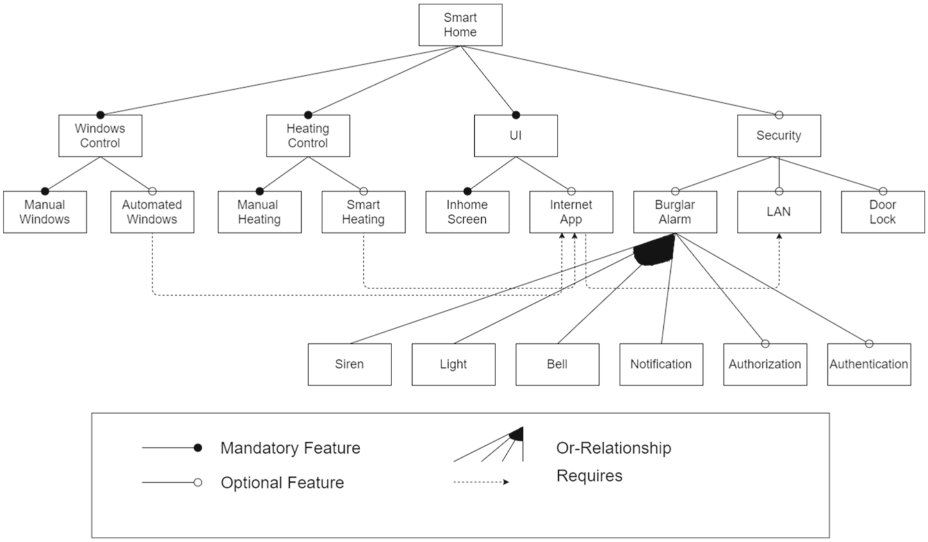


Fig. 1 Smart Home with 20 $\#features$

2.2 Related Work

Various researchers have found solutions to this reverse engineering problem. Haslinger et al. (2011) presented an algorithm that reverse engineers only one feature model and which identifies patterns in the selected and non-selected features. Later, he included Cross-Tree Constraints (CTCs) that requires and excludes as an extension in another work (Haslinger et al. 2013). Acher et al. (2012) came up with the proposal to map each feature into a feature model and build a single merged feature model. This merged feature model contains all the mapped features as feature models.

In article (Lopez-Herrejon et al. 2012), where reverse engineering of the feature models was done using Evolutionary Algorithms, two fitness functions were designed to be used. One is used to describe feature sets but disregards any surplus, and another one is used to get the desired number of feature sets.

Another approach (Acher et al. 2013) that investigated the synthesis of a feature model FM from a set of configurations used two main steps: characterise the different meanings of feature models and identify the key properties allowing to discriminate between them. The synthesis procedure was able to restitute the intended meanings of feature models based on inferred or user-specified knowledge.

Andersen et. al. (She et al. 2014) proposed algorithms for synthesis of feature models using the propositional constraints by deriving symbolic representations of all candidate diagrams, and deriving instances from this diagrams.

Three standard search based techniques (evolutionary algorithms, hill climbing, and random search) with two objective functions were evaluated for the reverse engineering task regarding feature models in paper (Lopez-Herrejon et al. 2015). The two objective functions were refined in a third one, an information retrieval measure.

In Assunção et al. (2016), authors used a set of feature sets to determine precision, recall and a dependency graph to compute variability safety of a feature model. This approach takes a multi-objective perspective and helps us to obtain feature models which represents the desired feature combinations and also make sure that these combinations are well-formed, more exactly variability safe.

Dynamic Software Product Line is another approach that is presented in article (Larissa Luciano Carvalho et al. 2017), but they do not address the problem of reverse engineering of feature models. The focus is on developing dynamically adaptable software that manages reconfiguration during execution. This approach aims to reuse components and also aims to adapt to environment changes or user requests, and is implemented with the help of Object Oriented Programming (OOP) as well as Aspect Oriented Programming (AOP).

We would like to emphasize that our replication is on a paper (Linsbauer et al. 2014) that is published earlier than some of the newer work on feature model synthesis that we mention above. The investigations (Lopez-Herrejon et al. 2015; Assunção et al. 2016; Larissa Luciano Carvalho et al. 2017) that follow this paper improve various aspects of the approach like: use of various standard search-based techniques with two objective functions for the reverse engineering task of feature models, multi-objective algorithms to obtain the desired feature combinations under consideration of variability safety, and developing dynamically adaptable software that manages reconfiguration during execution. The aim of our replication study is not to improve aspects of the proposed approach but to confirm the already obtained results and also to contribute information regarding the relation between number of features, number of generations and number of populations, providing insights into the best optimized values for several parameters of the genetic programming algorithm.

2.3 The Original Study

We conduct a replication study of the approach in paper (Linsbauer et al. 2014) where the authors applied genetic programming to the problem of reverse engineering feature models in the realm of SPLs. They started from a previous work (Lopez-Herrejon et al. 2012) where they used a genetic algorithm for reverse engineering feature models. In paper (Linsbauer et al. 2014), the authors started with a set of 17 feature models of actual SPLs. The implementation was made using ECJ framework [1] for evolutionary computation. For each of those feature models they computed the respective set of valid feature sets and used these sets as input for their Genetic Programming (GP) pipeline, Random Search (RS) baseline and genetic algorithm approach.

After obtaining the results, a statistical analysis was performed using the Wilcoxon Signed-Rank Test and the obtained results. The test was performed on the average fitness function values to compare the genetic programming approach against the random search baseline and also against genetic algorithm approach.

In the end, the test indicated a significant difference between those three approaches and by comparing the results it was established that genetic programming approach outperforms the genetic algorithm approach and the random search baseline.

The investigation considered the following GP parameters: Population size 100, Maximum number of generations 100, Crossover probability=0.7, Feature Tree Mutation probability=0.5, and CTCs Mutation probability=0.5.

To compare their results, as a base line, the authors used a Random Search (RS) that just randomly creates feature models in hopes of finding a good solution. The number of random tries is set to the product of the maximum number of generations and the population size of the GP. Thus the number of evaluated candidate feature model individuals is the same for both approaches that we consider (GP and RS), thus the number of performed evaluations is:

$maxGenerations \times populationSize = 100 \times 100 = 10000.$

The investigation examined also the results considering feature models with different number of features: 4 models with [6,10] features, 4 models with (10,15] features, 5 models with (15, 20] features, 4 models with $>$ 20 features.

The GP pipeline that was applied in the original study can also be found in our replication study. This GP pipeline includes a set of operators like Builder, Selection, Crossover, Reproduction, Mutation, Breeding that are next described.

2.3.1 Feature Model Representation

In the initial study, as well as in other studies, the Model Driven Engineering (MDE) approach was used. This approach uses a metamodel to define the structure and semantics of the models that can be derived from it. The metamodel is provided in Fig. 2. A simplified version of SPLX metamodel (which is actually a common representation for feature models) was chosen and describes the structure of a feature model individual. In the metamodel it can be seen that there is only one feature for the Root and for any other feature node. We

[1] http://cs.gmu.edu/~eclab/projects/ecj

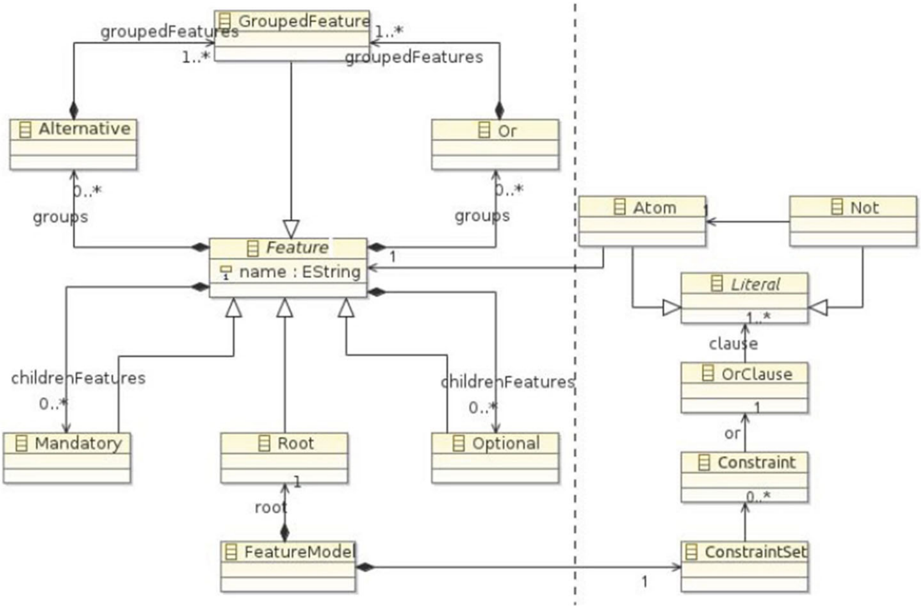


Fig. 2 Feature model metamodel (Linsbauer et al. 2014)

can also see the Mandatory and Optional child features as well as Alternative and Or group relations which must have at least one GroupedFeature as a child. Along with these are the CTCs of a feature model individual.

2.3.2 Fitness Function

This pipeline uses a fitness function based on information retrieval metrics for the Evaluator to describe the fitness of an individual. To obtain this function, the authors of the original study defined two auxiliary functions called Precision and Recall. According to the initial study, Precision is the fraction of the retrieved feature sets that are relevant to the search, and Recall is the fraction of the feature sets that are relevant to the search that are successsfully retrieved. Those are calculated using the following expressions:

$$precision(sfs, fm) = \frac{\#containedFeatureSets(sfs, fm)}{\#featureSets(fm)};$$

$$recall(sfs, fm) = \frac{\#containedFeatureSets(sfs, fm)}{|sfs|}.$$

The “$\#containedFeatureSets : SFS \times FM \rightarrow \mathbb{N}$” represents the number of feature sets received as first argument sfs that are valid according to a feature model fm. The “$\#featureSets : FM \rightarrow \mathbb{N}$” represents the number of feature sets denoted by a feature model fm. F$_\beta$ is a weighted measure of precision and recall. β indicates how many times the recall values weight more in comparison with the precision value. F$_\beta$ is obtained using the expression:

$$F_\beta = \frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall}.$$

Every node of a feature model is implemented as a function that manipulates a set of feature sets in order to compute the final feature sets that are represented by the whole feature model. This implementation is used to obtain the required metrics.

2.3.3 Operators, Mutation and Crossover

The necessary operators for GP were developed based on the tree structures that are derived from the metamodel and also on the domain constraints. Those necessary operators are Builder, Crossover and Mutator.

The Builder creates random feature trees and random CTCs according to the metamodel and to domain constraints. It was implemented in the oridinal study using the FaMa (Benavides et al. 2007) and BeTTy (Segura et al. 2012) frameworks.

A feature model individual suffers small random changes made by the Mutator. According to the initial study (Linsbauer et al. 2014), mutations that can be performed are:

- Randomly swaps two features in the feature tree.
- Randomly changes an Alternative relation to an Or relation or vice-versa.
- Randomly changes an Optional or Mandatory relation to any other kind of relation (Mandatory, Optional, Alternative, Or).
- Randomly selects a subtree in the feature tree and puts it somewhere else in the tree without violating the metamodel or any of the domain constraints.

Also, the mutations performed on the CTCs, that are applied with equal probability, are:

- Adds a new, randomly created CTC (i.e. clause) that does not contradict the other CTCs and does not already exist.
- Randomly removes a CTC (i.e. a clause).

The Crossover deals with the creation of two new individuals(offspring), from two individuals who belong to the current population (parents). Those new individuals should maintain desirable traits from both parents.

In the original study, how the crossover operator for feature model individuals works is described as follows:

- In the first step, the offspring is initialized with the root feature of Parent 1 and the root feature of Parent 2 is added to the offspring if is different from Parent 1. In this case, the root feature of Parent 2 is added as a mandatory child feature of its root feature.
- The second step is to cross the levels of the first parent, starting first from the root node, and add a random number r of features (that are not already contained) to the offspring. This is done by appending the features to their respective parent feature already contained in the offspring, using the same relation type between them.
- In the third step, the second step is repeated but this time using the second parent.
- In the last step is repeated the second step until every feature is contained in the offspring.
- The second offspring is obtained in the same way but starting with reversed positions of the parents. Regarding to the crossover for CTCs, the union of CTCs of the two

parents is performed and then it is assigned a random subset to the first offspring. The remaining is assigned to the second offspring.

2.4 Replication in Software Engineering

Experimentation plays a major role in scientific improvement, replication being one of the essentials of the experimental methods: experiments are repeated aiming to check their results. Successful replication increases the validity of the outcomes observed in the experiments.

For an experiment to be considered a replication (Shepperd et al. 2018), the following elements are required to be fulfilled: the authors must explicitly state which original experiment is being replicated; the purpose of the replication study includes extending the external validity of the experiment (i.e., adding to our understanding of how the results generalise); both experiments must have research questions or hypotheses in common (i.e., there are shared constructs and interventions), and the analysis must contain a comparison of original and new results with a view to conforming or disconforming the original experiment. Note that we intentionally avoid judgements such as “successful”. Internal (the replication team includes members from the original experiment) or external replications (the entire replication team is independent of the original team) may be performed: some researchers express a preference for external replications (being more independent but may be unintentionally less exact).

Other research investigations concerning how to do replications of software engineering experiments (Carver 2010; Carver et al. 2014) emphasized guidelines that suggest four types of information to include in a replication report: (1) information about the original study to provide enough context for understanding the replication; (2) information about the replication to help readers understand specific important details about replication itself; (3) comparison of replication results with original study results to illustrate commonalities and differences in the results obtained, and (4) conclusions across studies to provide readers with important insights that can be drawn from the series of studies that may not be obvious from a single study.

The elements of the software engineering experimental configuration are investigated and established in paper (Gómez et al. 2014). There are four dimensions: operationalization, population, protocol, and experimeters. A common conduct is that the authors should rigorously document their replication designs (Fagerholm et al. 2019).

Dimension: Operationalization. Operationalization describes the act of translating a construct into its manifestation, thus having cause and effect constructs: cause constructs are operationalized into treatments (they indicate how similar the replication is to the baseline expriment), and effect constructs are operationalized into metrics and measurement procedures.

Dimension: Population. Replications should also study the properties of experimental objects. Specifications, design documents, source code, programs are all examples of experimental objects. Replications examine the limits of the properties of experimental objects for which results hold.

Dimension: Protocol. The elements of the experimental protocol that can vary in a replication are: experimental design, experimental objects, guides, measuring instruments and data analysis techniques.

Dimension: Experimenters. This dimension refers to the people involved in the experiment. A replication (Gómez et al. 2014) should verify whether the observed results are independent of the experimenters by varying the people who performed each role (designer, trainer, monitor, measurer, analyst).

Each stated dimensions will be discussed in the next section, particularly for our replication design.

Replication in software engineering is of a paramount importance and must be conducted for both confirmatory and discovering additional knowledge of the investigated method.

2.5 Motivation for the Replication

Conducting a replication study (Dyba et al. 2005) has two main aims: first, to confirm the results of the original study, and secondly, to generate new knowledge that otherwise would not be possible to be created.

Thus, our replication further investigates the GP approach and the "behavior" of the algorithm when considering various $\#features$ with different characteristics, examining the relation between #features and #generations, and also between #features and #population, and also exploring the impact of GP parameters (crossover and mutation) on obtained solutions.

3 Research Design and Analysis

Our replication approach is detailed in this section, an overview of the replication and further, depicting the design of experiments. The replication experiments are detailed, stating the research questions and the used objects (i.e. feature models) and the steps in our replication design. Finally, this section presents the used data analysis, the Wilcoxon signed ranks test (Derrac et al. 2011) and the Taguchi method (Roy 2010).

3.1 Overview

We replicated the original study (Linsbauer et al. 2014) as an operational replication (Juristo and Vegas 2009) where we changed the replication objects (Gómez et al. 2014) and we partly varied the researchers (changed populations and experimenters). We used the original study protocol. This replication design addresses internal and external validity threats of the original study and adds new information regarding $\#generations$ and $\#populations$ of the GP approach, and also regarding the experiment parameters $\#crossover$ and $\#mutation$. It is also worth mentioning that the authors from the original study had here a different role, they do not conducted the experiments thus the experiments results were not biased considering this perspective. The original authors use their knowledge to leverage the knowledge in how to interpret the obtained results. The internal authors did not participate in the design and execution of the experiments. However, several meetings and discussions took place online about the original paper and experiments, about the current paper and the design of experiments, providing suggestions and validation of the proposed objectives. The external authors performed the execution of experiments, applied the Wilcoxon Signed-Rank Test and the Taguchi method.

The elements of the software engineering experimental configuration (Gómez et al. 2014), i.e. the four dimensions (operationalization, population, protocol, and experimeters) are next provided in the context of our replication design.

Dimension: Operationalization. Regarding *operationalization*, i.e. the cause constructs, our experimental replications are similar to the baseline experiment in that uses similar feature models with different number of features: many features models with 10 number of features and one for each set from (10, 15], (15, 20], and (20, 25].

Dimension: Population. With respect to *population*, the properties of experimental objects (i.e. feature models in our case) are considered when designing the experiments. Each experiment considered features models with various number of features.

Dimension: Protocol. With reference to *protocol*, some of the elements of the experimental protocol varied in the replication (i.e. other feature models with various number of features, Taguchi method for genetic programming's parameters optimization) and some other did not (i.e. measuring instruments using the same source code, and data analysis techniques, using Wilcoxon signed ranks test).

Dimension: Experimenters. With regard to *experimenters*, refering to the people involved in the experiment, we use a combination of researchers from the original study and also new researchers. We varied the people who performed each role (experiments designer, experiments execution and measurements, interpretation).

3.2 Goal Question Metric Model

This section details the research questions and the hypotheses. In order to better express and define our research questions, we use the Goal Question Metric (GQM) model (Basili and Rombach 1988). It helps to describe the necessary obligations for setting objectives before starting any software measurement activity. The GQM approach provides a framework involving several steps: list the major *Goals* of measurement; select a model that is connected to the measured entities; from each goal derive the *Questions* that must be answered to determine if the goal is met; decide what *Metrics*, based on the selected model, must be collected in order to answer the questions.

3.2.1 Goals

- G1: Confirm the original study results by using models with different numbers of features.
- G2: Determine relations between the number of features and the parameters of the genetic programming approach ($\#generations$, $\#population$).
- G3: Determine optimized parameter values for the genetic programming approach, i.e., $\#generations$, $\#population$, $\#crossover$ and $\#mutation$.

3.2.2 Questions

- RQ1: Can the original results of the original study be confirmed in other samples, i.e., with different feature models with similar characteristics (same values for the parameters $\#generations$, $\#population$, $\#crossover$ and $\#mutation$)?
- RQ2: Are there differences in the quality of the obtained solutions when using different values for the parameters $\#generations$ and $\#population$?
    - RQ2a) What is the relation between Number of Features and Number of Generations ($\#generations$)?
    - RQ2b) What is the relation between Number of Features and Number of Individuals ($\#population$)?

- RQ3: Do values for crossover and mutation play a role in obtaining better solutions?

The first research question targets the replication of the original experiments and is thus a *confirmatory replication*. The second and third questions are integrated into the *exploratory replication* aiming to yield new knowledge about the used method.

3.2.3 Metrics

We use the following four metrics to answer the three research questions.

- M1: *Paired Difference* (via Wilcoxon signed-rank test).
- M2: *Mean Fitness Values* of the obtained solutions for GP and RS approaches.
- M3.1: *Mean of Means* (via L4 Taguchi method)
- M3.2: *Mean of Means* (via L9 Taguchi method, 3 trials)

The first research question will be investigated and analyzed by applying multiple experiments with different models with 10 features. We then determine if there is a statistically significant difference (metric M1) between the results of the original study and the replicated results to answer the first research question.

The second research question will be explored by performing experiments with algorithms run with varying number of features and different values for parameters $\#generations$ and $\#population$.

The third research question is addressed by applying the Taguchi method for the design of experiments in order to optimize the GP parameters. We investigate the optimized values for the following parameters: $\#generations$ and $\#population$, as well as $\#mutation$ and $\#crossover$.

3.3 Experimental Design, Experimental Objects and Analysis

This section provides details relating to the experiments: the key elements regarding the design of the experiments, the objects used in the experiments, and the methods used to analyze the results.

3.3.1 Experimental Design

We conducted a replication study on the paper (Linsbauer et al. 2014) using various experiments, i.e. three sets of experiments each consisting of other experiments.

Figure 3 sketches the design of our replication experiments: Experiments Set 1) - simple replication (duplicate the exact conditions), Experiments Set 2) - interaction of 2 parameter of the GP approach, and Experiments Set 3) - experiments with aim to optimize the experimental parameters number of generations, population, crossover and mutation.

We aim to both confirm the original obtained results (thus to produce generalized knowledge) and also to yield additional knowledge by finding out more about the range of conditions under which the results hold.

Our first attempt to replicate the approach (named *Experiments Set 1)*) used three experiments with the following mentioned characteristics:

- Experiment 1.1) 10 models with 10 features;
- Experiment 1.2) 5 models with [5, 10, 15, 20, 25] features;
- Experiment 1.3) one model with 20 features with various number of generations (50, 75, 100, 300, 500, 2500).

The experiments were designed as discussed by Yin (2008). The 1.1) experiment was done for 10 models, each with 10 features. This experiment was designed as a "simple" replication discussed by Yin: "Some of the replications might attempt to duplicate the exact conditions of the original experiment". The design of the other experiments followed the procedure of Yin regarding experiments design, i.e. altering "one or two experimental conditions considered unimportant to the original finding" with the aim to see if the finding could still be duplicated. The 1.2) experiment aimed to investigate how the approach behaved when increasing the number of features. The 1.3) experiment investigates how the algorithm behaves when various numbers of generations are used.

Starting from those primary experiments we further investigated deeper aspects of the approach parameters: Experiment 1.2) and Experiment 1.3) are further extended. The Experiment 1.2) is extended such that the algorithms will be run with the following configurations: do the experiments also for 50, 75, 300, 500, 1000 regarding $\#population$, forming thus Experiment 2.2). The Experiment 1.3) is extended such that the algorithms will be run with the following configurations (do the experiments also for 1000 regarding $\#generations$ and also for: 15, 25 regarding $\#features$), forming thus Experiment 2.1).

Thus, these set of experiments form *Experiments Set 2)* of this replication study, as follows:

- Experiment 2.1) 5 models with [5, 10, 15, 20, 25] features with various number of generations [50, 100, 300, 500, 1000];
- Experiment 2.2) 5 models with [5, 10, 15, 20, 25] features with various number of population (50, 100, 300, 500, 1000).

We also aim to determine the optimized values for the parameters of the genetic programming approach, i.e. $\#generations$, $\#population$, $\#crossover$ and $\#mutation$. Therefore, these set of experiments form *Experiments Set 3)* of this replication study, as follows:

- Experiment 3.1) L4 Taguchi method to optimize the $\#population$ and $\#generations$ parameters for the genetic programming approach.
- Experiment 3.2) L9 Taguchi method to optimize the following parameters for the genetic programming approach: $\#population$, $\#generations$, $\#crossover$ and $\#mutation$.

Details about the design of the third experiments set are provided in Section 4.1.3.

No experiments were performed in the original paper to investigate best parameter values for the genetic programming algorithm. Thus, similar experiments to Experiments set 2 and Experiments set 3 were not performed and discussed in the original paper. We aim to obtain concrete best values for several parameters: number of generations and the population number, the mutation and crossover parameters.

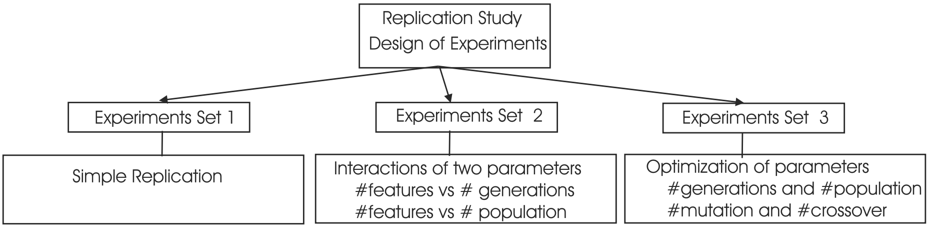

Replication Study
Design of Experiments
Experiments Set 1
Experiments Set 2
Experiments Set 3
Simple Replication
Interactions of two parameters
#features vs # generations
#features vs # population
Optimization of parameters
#generations and #population
#mutation and #crossover

Fig. 3 Our design of experiments for the replication study

Table 1 Feature models with 10 features (source SPLOT (http://www.splot-research.org/))

Feature model name	Number of features
Electric car	10
Example mobile phone	10
Maquina de café	10
Minimalistic PC feature model	10
Mobile media	10
Smart home	10
Smartwatch	10
TV	10
E-SHOP-MM	10
E-SHOP	10

3.3.2 Experimental Objects

Our experiments considered feature models with various number of features. We provide the set of used feature models in our experiments in the replication package.[2] The first experiment considered 10 different feature models, each of them having 10 features. Table 1 contains the name of the used feature models taken from SPLOT (http://www.splot-research.org/).

For the second set of conducted experiments, we used various versions of the same feature model, but modifying the number of features. The basic feature model is from the source (http://www.splot-research.org/). One of the experiments considered varying the $\#generations$ (while varying also $\#features$) and the other experiment considered varying the $\#population$ (while varying also $\#features$). Different $\#features$ were used: 5, 10, 15, 20, 25. For $\#generations$ the following values were used: 50, 100, 300, 500, 1000 while for $\#population$ we used: 50, 100, 300, 500, 1000.

3.3.3 Analysis of Experiments

For the analysis of the obtained results we have used the Wilcoxon signed ranks test (Derrac et al. 2011) and the Taguchi method (Roy 2010). This section describes them.

When comparing (Harman et al. 2012) two algorithms, the best fitness values obtained by the searches concerned are an obvious indicator to how well the optimisation process performed. Inferential statistics may be applied to discern whether one set of experiments are significantly different in some aspect from another. Usually we wish to be in a position to make a claim that we have evidence that suggests that the GP algorithm is better than RS algorithm.

Wilcoxon Signed Ranks Test The Wilcoxon signed ranks test (Derrac et al. 2011) is used for answering the following question: do two samples represent two different populations? It is a nonparametric procedure employed in hypothesis testing situations, involving a design with two samples. It is a pairwise test that aims to detect significant differences between two sample means, that is, the behavior of two algorithms. The best fitness value (from the entire population) was used for comparing the two algorithms.

[2] https://www.cs.ubbcluj.ro/~avescan/publications/ReplicationPackageEMSE.zip

The Wilcoxon signed ranks test has two hypothesis:

1. Null hypothesis H_0: The median difference is zero versus.
2. Research hypothesis H_1: The median difference is not zero, $\alpha = 0.05$.

The steps of the Wilcoxon signed ranks test are: compute W_- (sum of the negative ranks) and W_+ (sum of the positive ranks); check if $W_- + W_+$ = n(n+1)/2; select the test statistic (for the two tailed test the test statistic is the smaller of W_- and W_+); we must determine whether the observed test statistic W_t supports the H_0 or H_1, i.e. we determine a critical value of W_c such that if the observed value of W_t is less or equal to critical value W_c, we reject H_0 in favor to H_1.

Due to stochastic nature of optimisation algorithms, searches must be repeated several times in order to mitigate against the effect of random variation. How many runs do we need when we analyze and compare algorithms? In many fields of science (i.e. medicine and behaviour science) a common rule of thumb (Arcuri and Briand 2011) is to use at least $n = 30$ observations. We have also used in our evaluation 30 executions for each algorithm. We have also used Wilcoxon statistical test to compare the results of the GP and RS algorithms as in the original study.

Taguchi Method Design Of Experiments (DOE) (Roy 2010) is known as the technique of defining and investigating all possible conditions in an experiment that involves multiple factors, technique known also as factorial design.

The relative influence of the factors and interactions between factors included in the study can be quantitatively determined using the analysis of variance. For a full factorial design, the number of possible designs, N, is $N = L^m$, where L=number of levels for each factor and m=number of factors.

For small values of L and m, for example L=2 and m=3, there are 2^3 possible design configurations and the analysis could be done with reasonable resources, both money and time. But when having 15 factor, even with only two levels, a market research for this study would be unreasonable regarding used resources.

Fractional factorial experiments investigate only a fraction of all possible combinations but there are several limitations as stated in Roy (2010).

Dr. Genichi Taguchi proposed an innovative method of simplifying and standardizing fractional factorial designs. Details about the method could be found in Roy (2010), describing the standardized DOE, the robust design strategy and signal-to-noise analysis (from multiple-sample tests).

The Taguchi method has the main goal to find optimal parameters to reduce the randomness in stochastic solution-search style algorithms. When the performance and quality of a product or manipulation are not effected by internal and external disturbances, we recognize it as the effect of robustness. The quality characteristics (Phadke 1989) of a produced product are affected by the noise factors X and control factors Z, and the output results Y maybe deviate from the expected target M. The quality characteristics are changed by the noise factors, so the quality deviates from the target values as represented in Fig. 4. In order to achieve the expected target, the control factors are adjustable parameters. When the Taguchi method is used with genetic programming methods, the control factors are the population size, the number of generations, crossover rate, and mutation rate.

Various approaches used the Taguchi methods to tune parameters of multi-objective evolutionary algorithms. The approach in Sadeghi and Niaki (2015) investigates number of

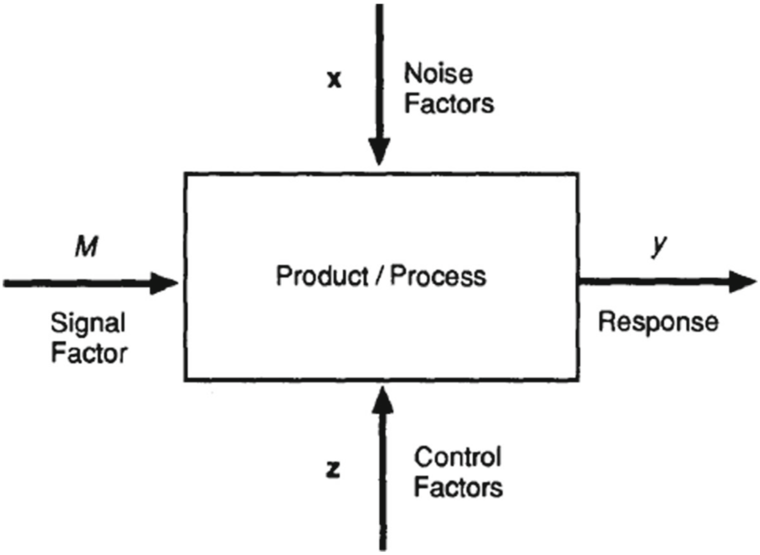


Fig. 4 Effect of control factors and noise factors on quality characteristics (Phadke 1989)

iterations, number of population, probability for crossover and mutations. The Parwananta et al. (2013) approach conducted a factorial experiment for three parameters (population size, crossover and mutation rates) and established them for their two-phase genetic algorithm for Solving the Paired Single Row Facility Layout Problem.

We have also used the Taguchi methods to obtain the optimized parameters for both $\#generations$ and $\#population$ using $L4$ design (two factors with two level each), and also $L9$ design adding the $\#crossover$ and $\#mutation$ values, with three level each. For the $L4$ design we thus considered $\#population$ and $\#generations$ with values 100 and 300 for each, and for the $L9$ design we considered $\#population$, $\#generations$ $\#crossover$ and $\#mutation$ with values 100, 200 and 300 for $\#population$ and $\#generations$, for $\#crossover$ the values 0.70, 0.75, 0.80 and for $\#mutation$ the values 0.50, 0.60 and 0.70.

4 Replication Results

The steps for experiments executions and the obtained results are provided in this section. The analysis of the hypotheses and a comparison between original and replicated studies is scrutinized.

4.1 Experiment Execution

The results of the experiments executions are provided and discussed next. As explained in Section 3.1 the three sets of experiments were design with different purposes: simple replication that considered similar features models, the second experiments set investigated the relations between various characteristics of feature models (number of features and number of generations and population), and the last one with the aim to optimize parameters of the genetic programming algorithm.

Table 2 Replication Study of the original study

Paper	#FM used	[5,10],	(10,15]	(15,20]	(20, 25]
Original study	17	4	4	5	4
This study	14	10	1	1	1

4.1.1 Results of Experiments Set 1

The first set of experiments concerned a simple replication of the original study. Three different experiments were conducted, each aiming to replicate the original experiments considering to vary/expand different elements: experiment with models with 10 features, experiment with models with different number of features, and experiment for the same model but with different number of generations. Table 2 contains the number of features models used in the original study and in the replicated study, referring to the number of contained features. The first experiment in the first set replicates considering 10 models with 10 features, the second and the third experiments consider 5 features models with different number of features (from 5 to 25).

The first experiment in the first set 1.1) considered 10 models with 10 features each. This experiment was designed as a "simple" replication discussed by Yin (2008): "Some of the replications might attempt to duplicate the exact conditions of the original experiment." For this experiment we considered $\#population$ size of 100 and $\#generations$ to be 100, as in the original study.

Table 3 presents the obtained results. The results of the Wilcoxon statistical test that compared the results of the GP and RS algorithms concluded that the null hypothesis is rejected and alternative hypothesis is accepted, thus our findings confirmed the results from the original paper. We restate here that the best fitness value (from the entire population) was used for comparing the two algorithms using the statistical test.

The second experiment in the first set 1.2) considered five different models with various number of features (5, 10, 15, 20, 25). The considered models used as a basis the SmartHome model, in Fig. 1 the model with 20 features being presented.

Table 4 contains the obtained results for the Wilcoxon statistical test (the best fitness value from the entire population was used) for the five features models with various number of features: the null hypothesis is rejected and alternative hypothesis is accepted, thus our findings confirmed the results from the original paper.

Table 3 Fitness based Wilcoxon Signed-Rank Test for Experiment 1.1) with 10 models and 10 features each

Model	W_+	W_-	W_{test}	N	W_{critic}	H_0	H_1
Electric car	0	465	0	30	120	×	✓
Example mobile phone	0	465	0	30	120	×	✓
Maquina de café	0	465	0	30	120	×	✓
Minimalistic PC feature	0	465	0	30	120	×	✓
Mobile media	0	465	0	30	120	×	✓
Smart home	0	465	0	30	120	×	✓
Smartwatch	0	465	0	30	120	×	✓
TV	0	465	0	30	120	×	✓
E-SHOP-MM	0	465	0	30	120	×	✓
E-SHOP	0	465	0	30	120	×	✓

Table 4 Fitness based Wilcoxon Signed-Rank Test for Experiment 1.2) with 5 models with various features

Model	W_+	W_-	W_{test}	N	W_{critic}	H_0	H_1
5 features Smart Home	0	465	0	30	120	×	✓
10 features Smart Home	0	465	0	30	120	×	✓
15 features Smart Home	0	465	0	30	120	×	✓
20 features Smart Home	0	465	0	30	120	×	✓
25 features Smart Home	0	465	0	30	120	×	✓

The third experiment in the first set 1.3) examined for the same model what is the behavior of the algorithm when using different number of generations: 50, 75, 100, 300, 500, 2500.

Table 5 contains the obtained results for the Wilcoxon statistical test (using the best fitness value from the entire population) for the Smart Home features model with 20 features but considering various number of generations: the null hypothesis is rejected and alternative hypothesis is accepted, thus our findings confirmed the results from the original paper.

Together, the three experiments in the first set confirm the findings in the original study, i.e. genetic programming provides better reverse engineering results than random search.

4.1.2 Results of Experiments Set 2

This set of experiments investigate the relation between number of features and the number of generations, and between number of features and the number of population used in the algorithm, i.e. how is the relation between those parameters.

The first experiment from this set 2.1) considered investigating the relation between the number of features and the number of used generations. Figure 5 displays the results of the Experiments Set 2.1), comparing various generations versus various features for the GP and RS algorithms.

It can be noticed that starting from 10 features the GP approach has better results than the RS approach, i.e. the average GP fitness values obtained being better than those of RS. This remark is true for all considered number of generations used.Thus, for each category of feature models (from 5 to 25 features) the GP values are higher than the RS values.

It should also be stated that starting from 15 features, there is a larger gap between the quality of the solutions obtained with 100 generations versus 300 generations. And, there is a clear observation that by increasing the number of generations, the quality of the solutions also increases. For example, at the last experiment (25 features) there is an increase of the quality from 50 to 1000 generations.

The second experiment from this set 2.2) considered investigated the relation between the number of features and the population number.

Table 5 Fitness based Wilcoxon Signed-Rank Test for Experiment 1.3) with the model with 20 features and varying number of generations

Model-SmartHome-20features	W_+	W_-	W_{test}	N	W_{critic}	H_0	H_1
50 generations	0	465	0	30	120	×	✓
75 generations	0	465	0	30	120	×	✓
100 generations	0	465	0	30	120	×	✓
300 generations	0	465	0	30	120	×	✓
500 generations	0	465	0	30	120	×	✓
2500 generations	0	465	0	30	120	×	✓

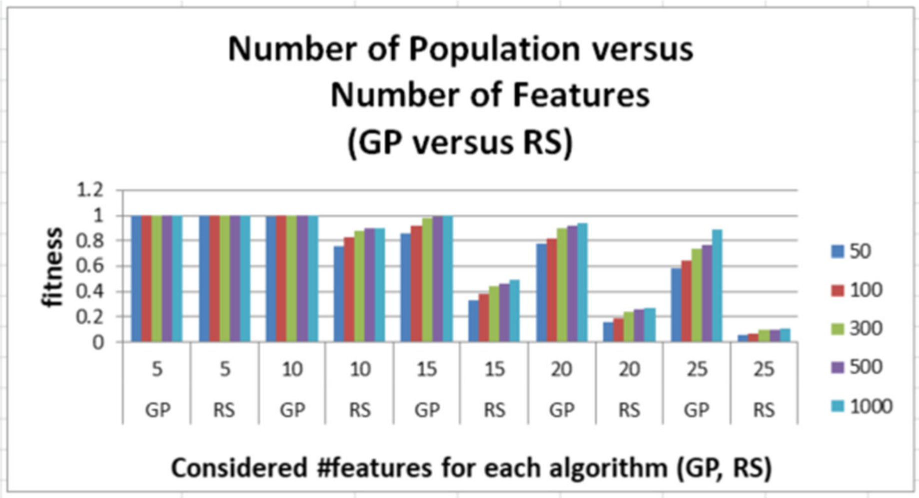


Fig. 5 Generations versus Features for the two considered algorithms (GP and RS)

Figure 6 presents the results of the Experiment Set 2.2), comparing various population numbers versus various features for the GP and RS algorithms.

It can be noticed that starting from 10 features the GP approach has better results than the RS approach, i.e. the average GP fitness values obtained are better than those of RS. This remark/conclusion/statement is true for all considered numbers of population used. Thus, from the experiment with 10 features to the experiment with 25 features, the GP solution achieves higher fitness values than the RS solution.

It should also be stated that starting from 15 features, there is a larger gap between the quality of the solutions obtained with 100 population number versus 300 population number. And, there is a clear observation that by increasing the number of population, the quality

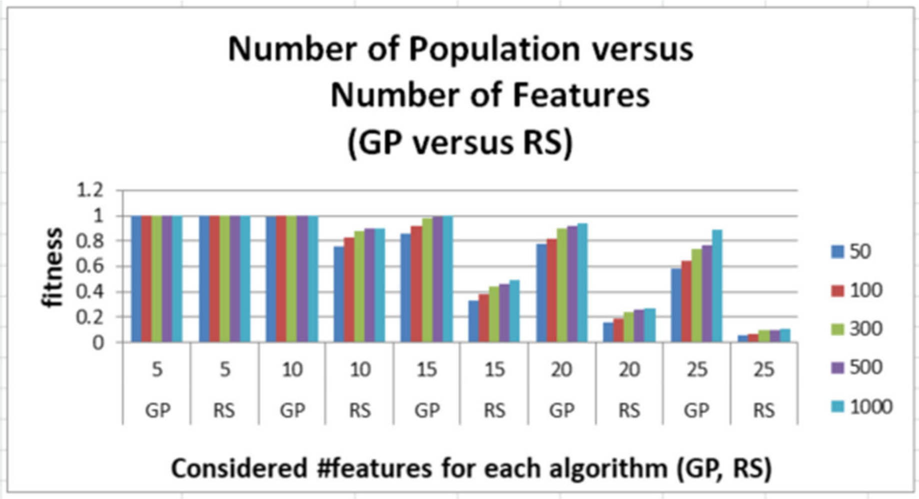


Fig. 6 Population versus Features for the two considered algorithms (GP and RS)

of the solutions increase also. For example, at the last experiment (25 features) there is an increase of the quality from 100 to 300 population.

A further novel finding regarding using genetic programming for reverse engineering feature models concerns the values for the $\#generations$ and $\#population$, values obtained empirically by conducting various experiments. There is a difference in the quality of the obtained solutions with 100 generations versus 300 generations, respectively solutions obtained with 100 population versus 300 population.

4.1.3 Results of Experiments Set 3

For this set of experiments we have used the Taguchi method to optimize the parameters: L4 for the $\#population$ and $\#generations$ and also L9 Taguchi method for the $\#population$, $\#generations$, $\#crossover$ and $\#mutation$ parameters with three trials.

L4 Taguchi Method We have conducted the Taguchi method to optimize the $\#population$ and $\#generations$ parameters. We have considered two levels for each of the factors, i.e. 100 and 300 as provided in Table 6 for L4 design configuration. For these experiments we considered one execution of the genetic programming algorithm.

We have applied the method considering three different type of features models: with 15 features, 20 features and 25 features. Figures 7, 8 and 9 contains the results of the Taguchi L4 design for features models with 15, 20 and 25 features models: we plot the main effects for mean. A line connects the points for each factor. Because the line is not horizontal, there is a main effect. Different levels of the factor affect the characteristic differently. The best value by $\#population$ and $\#generations$ obtained in all experiments is 300, for both parameters. Inspecting Figs. 7, 8 and 9 we may notice that, on average, experimental runs with $\#population$ and $\#generations$ of 300 had higher signal-to-noise ratio than experimental runs with 100.

L9 Taguchi Method We have conducted the Taguchi method to optimize the $\#population$, $\#generations$, $\#crossover$ and $\#mutation$ parameters. We have considered three levels for each of the factors. Table 7 contains the used configuration for the L9 design.

We have applied the method considering three different type of features models: with 15 features, 20 features and 25 features and considering three trials (experiments of the algorithm). For these experiments to analyze the results we use signal to noise ration (S/N) that is used for experiments with multiple runs.

Figures 10, 11 and 12 contain the results of the Taguchi L9 design for features models with 15, 20 and 25 features models. A line connects the points for each factor. Because the line is not horizontal, there is a main effect. When the line is horizontal, there is no main effect: each level of the factor affects the characteristic in the same way. The best values for $\#population$ is 300 for reverse engineering features models with 15 features and 25

Table 6 Taguchi L4 method applied for the $\#population$ and $\#generations$ parameters with two levels: 100 and 300

Number	$\#population$	$\#generations$
1	100	100
2	100	300
3	300	100
4	300	300

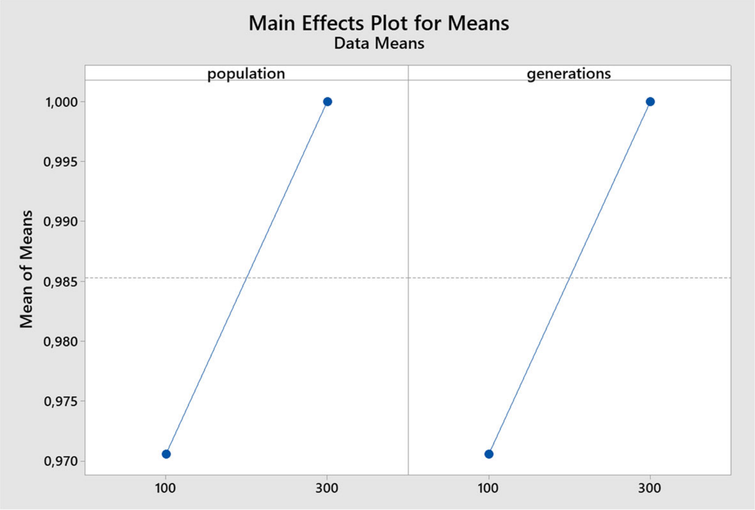


Fig. 7 Results of Taguchi L4 design for Feature Models with 15 features

features, but 200 for features models with 20 features:in the first column in all three Figs. 10, 11 and 12 the "highest" point. Regarding the optimized value for the *#generations* (second column in all three Figures), we obtained the same value, i.e. 300, regardless the number of features considered. For *#crossover* (third column) the optimized value is 0.75 starting with feature models that have 20 features. For *#mutation* (fourth column in all three Figures) is different depending on the type of the feature model (0.7 for feature models with 15 and 20 features, and 0.6 for feature model with 25 features).

Inspecting Figs. 7, 8 and 9 we may notice that, on average, experimental runs with *#population* and *#generations* of 300 had higher signal-to-noise ratio than experimental runs with 100.

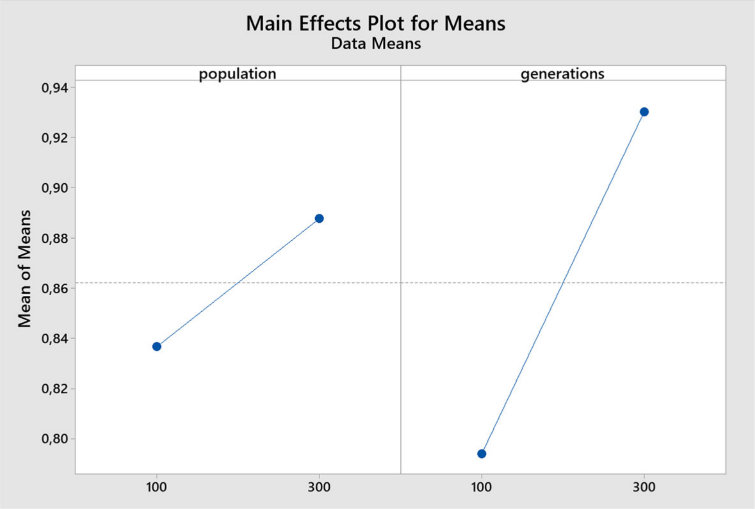


Fig. 8 Results of Taguchi L4 design for Feature Models with 20 features

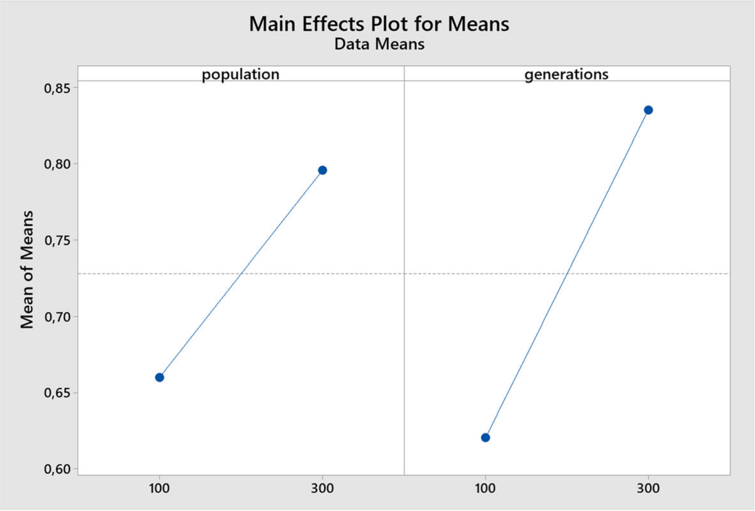


Fig. 9 Results of Taguchi L4 design for Feature Models with 25 features

The results of the experiments found clear support for the optimized values for various investigated parameters: $\#population$, $\#generations$, $\#crossover$ and $\#mutation$. The optimized value for the number of generations is 300, whereas for the number of population it varies from 200 to 300 for features models with different number of features. Regarding $\#crossover$, the results discovered that starting from 20 features, the best value is 0.75. With respect to $\#mutation$, for feature models with 15 and 20 number of features the best value is 0.70, whereas for feature models with 25 features the best values is 0.6. As observed from Figs. 10, 11 and 12, there is a clear distinction for the better values regarding $\#population$ and $\#generations$, i.e. from a flat level at 200 and 300 to an elevated value to 300 starting with models with 20 features. For $\#crossover$ the result is stabilized with the 20 features and 25 features experiments, i.e. the best obtained value is 0.75. Regarding $\#mutation$, while the number of used features in the experiments increase we obtained smaller best results for this parameter, i.e. 0.7 for the experiment with 20 features and 0.6 for the experiment with 25 features.

4.2 Research Questions Analysis

This section answers to each of the considered Research Questions using the results of the conducted experiments.

Table 7 Taguchi L9 method applied for the $\#population$, $\#generations$, $\#crossover$ and $\#mutation$ parameters with three levels

Parameter	Level 1	Level 2	Level 3
$\#population$	100	200	300
$\#generations$	100	200	300
$crossover$	0.70	0.75	0.80
$mutation$	0.50	0.60	0.70

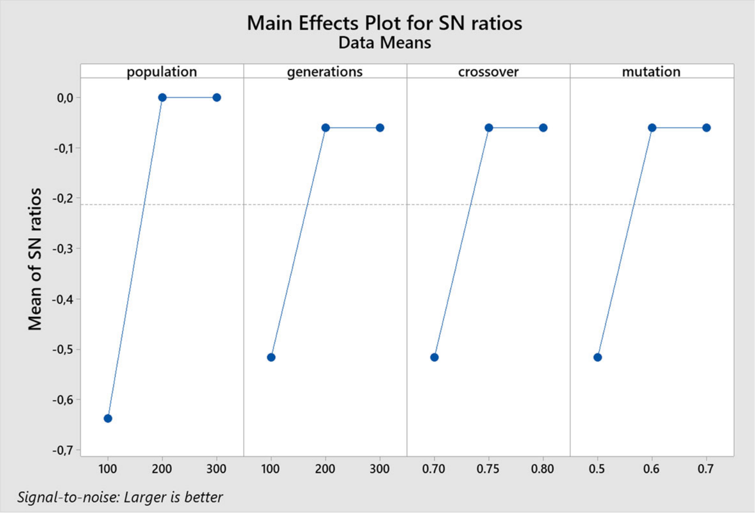


Fig. 10 Results of Taguchi L9 design for Feature Models with 15 features

The first research question that we answer uses the obtained results in the conducted experiments in set 1, results presented in Section 4.1.1.

RQ1: Is the original results in the original study confirmed in other samples?

All the experiments rejected the null hypothesis, thus the alternative hypothesis is accepted. Our findings confirmed the results from the original paper, i.e. genetic programming finds better solutions than the random search approach for reverse engineering feature models.

The second research question that we answer uses the obtained results in the conducted experiments in set 2, results presented in Section 4.1.2.

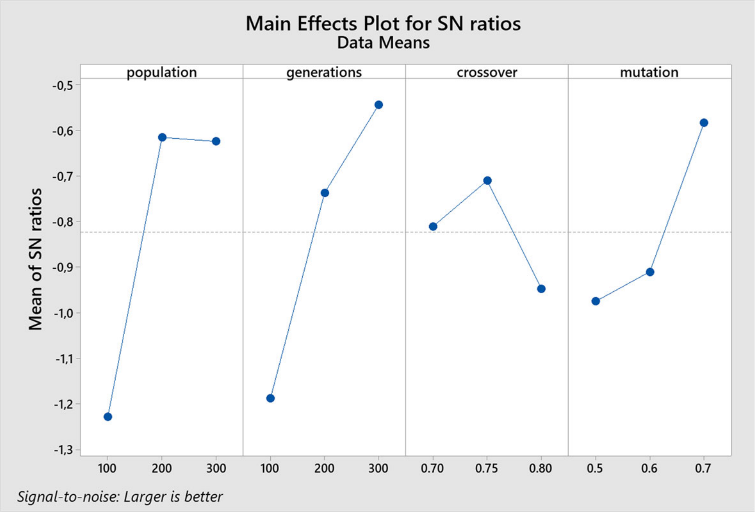


Fig. 11 Results of Taguchi L9 design for Feature Models with 20 features

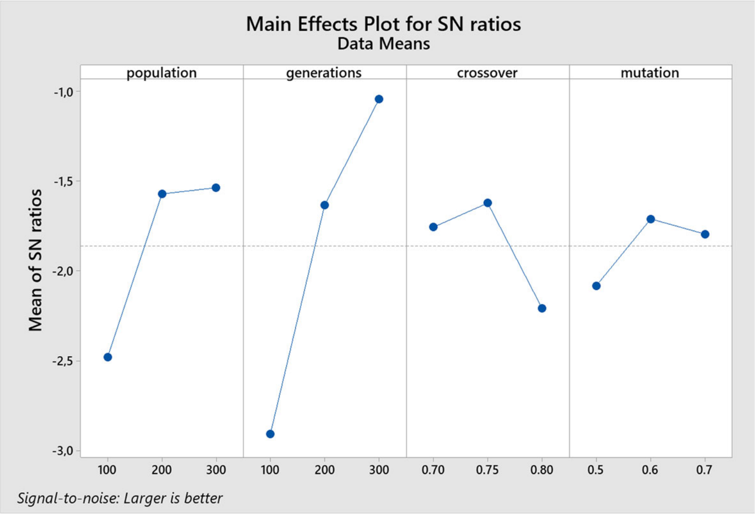


Fig. 12 Results of Taguchi L9 design for Feature Models with 25 features

RQ2: Are there differences in using various numbers for $\#generations$ and $\#population$?

RQ2a) What is the relation between Number of Features and Number of Generations?

RQ2b) What is the relation between Number of Features and Number of individuals (population)?

The results obtained in the second set of experiments establish that the GP approach obtains better results compared to RS approach. Moreover, there is an increment of the obtained solution quality from 100 to 300 for both $\#generations$ and $\#population$, experimenting with $\#features$ larger than 10.

The third research question that we answer uses the obtained results in the conducted experiments in set 3, results presented in Section 4.1.3.

RQ3: Do values for crossover and mutations play a role in obtaining better solutions?

The conducted experiments with the Taguchi method for both L4 and L9 designs revealed that best value for $\#generations$ is 300, for $\#population$ varies from 200 to 300 based on the number of features, and that for $\#crossover$ is 0.75 (starting from feature models with 20 features), whereas for $\#mutation$ the best value is 0.7, but starting with 25 features the best value is 0.6.

4.3 Result Comparison Between Original and Replicated Studies

The results of this replication remains consistent with the results of the original study. Therefore we build knowledge to support the original study findings and also provide new information about the conditions and parameters of the applied method.

This study has validated the result of the original study by executing similar feature models under the same conditions. The results confirmed the findings in the original study: the genetic programming algorithm for reverse engineering feature models is better than the random search algorithm considering feature models with various number of features.

On the other hand, the knowledge built by this study refers to the optimized values for several parameters of the genetic programming approach. Specifically, we found the following values: for #$generations$ the value is 300, for #$population$ the values are different depending on the number of features, so either 200 or 300, for #$crossover$ the best value is 0.75 starting with 20 number of features, whereas for #$mutation$ the value is 0.70 around 20 features and 0.60 around 25 features.

5 Threats to Validity

In this section, we address potential threats to the validity of the study and discuss some bias that may have affected the study results. We also explain our actions to mitigate them.

5.1 Internal Validity

We have identified six threats regarding internal validity. In what follows, we detail them and discuss how we mitigated them.

One threat to internal validity is the selection of the feature models which are relevant to our approach. Although some of these selected feature models come from the same domain, they are selected from populations with different characteristics. They have different number of features and different number of Cross-Tree Constraints.

The second threat to internal validity refers to the algorithm used to compare our approach. The Random Search baseline randomly creates feature models. Thus the population may be again affected.

Another threat to internal validity is maturation. In combination with the number of population, number of generations and number of features, the quality of the solutions also increases. The replication study did not investigate execution time. As there is a known trade-off in GP between quality versus time, this is another threat.

The correctness of the implementation is an internal validity threat that has been addressed by using the implementation and the data that were made available for replication.

With regard to experimenters, refering to the people involved in the experiment, we use a combination of researchers from the original study and also new researchers. We varied the people who performed each role (experiments designer, experiments execution and measurements, interpretation).

The internal validity threat referring to the overlap of authors between the original study and the replication study was addressed by assigning different roles to the authors of the original study: they did not participate in the design and execution of the experiments, rather we leveraged their knowledge on how to interpret the obtained results. The external authors performed the execution of experiments, applied the Wilcoxon Signed-Rank Test and the Taguchi method.

5.2 External Validity

With regard to external validity, we have distinguished four threats that are detailed next.

The selection of participants can be a threat to external validity. However, the feature models selected by us proved to be relevant to confirm the initial study.

The variation of the parameters for the algorithms can be another threat to external validity. By using other parameters one could obtain feature models with other characteristics which are not favorable. For the first set of experiments we used the same values that were used in the original study (Linsbauer et al. 2014). For the second and third experiments sets we used different values for generations, population, mutation and crossover.

Regarding the chosen populations for the Taguchi experiments, only three population values were investigated in the experiments: 100 and 300, respectively 100, 200, 300. With the largest value we obtained the best solutions, thus leaving open the possibility that increasing the population size further will improve the results further.

Lack of evaluations for instances of growing size and complexity is also threat. By using feature models with different number of features, varying in size and complexity, we tried to address this problem.

5.3 Construct Validity

Linked to construct validity, guessing the hypothesis is a threat. Being a replication study we can say that for the first set of experiments we expected the null hypothesis to be rejected. Even if our assumption proved to be true for the first set, we can't say the same for the second and third set of experiments due to the variation of the parameters.

6 Conclusion

Building a system with a large set of client requirements by reverse engineering feature models is time-consuming and requires considerable effort from the developers. Automated support comes thus to reduce the effort and time.

One of the approaches that produced good reverse engineering results of feature models is genetic programming. We proposed a replication of this method, using a larger set of feature models for the exact replication conditions, while also investigating various values of the genetic programming parameters and operators.

Three perspectives were integrated in the design of our replication experiments: duplicate the exact conditions using various features models, study the interaction of two parameters of the genetic programming approach, and optimization of the values for the four operators.

Results emphasize the previously obtained outcome, the original study being confirmed. The investigation also yields additional knowledge, finding out more about the range of conditions under which the results hold. There are relations between the number of features and number of generations, respectively number of features and number of population. Also, optimization of various parameters of the genetic programming approach was performed.

Acknowledgements We thank the anonymous reviewers for their careful reading of our manuscript and their many insightful comments and suggestions.

Funding Open access funding provided by Johannes Kepler University Linz.

References

Software product lines online tools. http://www.splot-research.org/. Accessed: 18 May 2019

Acher M, Baudry B, Heymans P, Cleve A, Hainaut JL (2013) Support for reverse engineering and maintaining feature models. In: Proceedings of the Seventh international workshop on variability modelling of software-intensive systems, VaMoS '13. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2430502.2430530

Acher M, Cleve A, Perrouin G, Heymans P, Collet P, Lahire P, Vanbeneden C (2012) On extracting feature models from product descriptions. https://doi.org/10.1145/2110147.2110153

Arcuri A, Briand L (2011) A practical guide for using statistical tests to assess randomized algorithms in software engineering. In: International conference on software engineering, pp 1–10

Assunção W, Lopez-Herrejon R, Linsbauer L, Vergilio S, Egyed A (2016) Multi-objective reverse engineering of variability-safe feature models based on code dependencies of system variants. Empir Softw Eng, 22. https://doi.org/10.1007/s10664-016-9462-4

Basili V, Rombach D (1988) The tame project: towards improvement-oriented software environments. IEEE Trans Softw Eng 14(6):758–773

Benavides D, Segura S, Trinidad P, Ruiz-cortés A (2007) Fama: Tooling a framework for the automated analysis of feature models. In: Proceeding of the first international workshop on variability modelling of softwareintensive systems (VAMOS, pp 129–134

Carver JC (2010) Towards reporting guidelines for experimental replications: a proposal. In: The international workshop on replication in empirical software engineering, pp 2–5

Carver JC, Juristo N, Baldassarre MT, Vegas S (2014) Replications of software engineering experiments. Empir Softw Eng 19(2):267–276. https://doi.org/10.1007/s10664-013-9290-8

Derrac J, Garcia S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation 1:3–18

Dyba T, Kitchenham BA, Jorgensen M (2005) Evidence-based software engineering for practitioners. IEEE Softw 22(1):58–65

Fagerholm F, Becker C, Chatzigeorgiou A, Betz S, Duboc L, Penzenstadler B, Mohanani R, Venters CC (2019) Temporal discounting in software engineering: a replication study. In: 13Th ACM/IEEE international symposium on empirical software engineering and measurement, IEEE, pp 1–12. https://doi.org/10.1109/ESEM.2019.8870161

Gómez OS, Juristo N, Vegas S (2014) Understanding replication of experiments in software engineering: a classification. Inform Softw Technol 56(8):1033–1048. https://doi.org/10.1016/j.infsof.2014.04.004

Harman M, McMinn P, de Souza JT, Yoo S (2012) Search based software engineering: Techniques, taxonomy, tutorial. Empirical Software Engineering and Verification 7007:1–59

Haslinger EN, Lopez-Herrejon RE, Egyed A (2011) Reverse engineering feature models from programs' feature sets. In: 2011 18Th working conference on reverse engineering, pp 308–312. https://doi.org/10.1109/WCRE.2011.45

Haslinger EN, Lopez-Herrejon RE, Egyed A (2013) On extracting feature models from sets of valid feature combinations. In: Cortellessa V, Varró D (eds) Fundamental approaches to software engineering. Springer, Berlin, pp 53–67

Juristo N, Vegas S (2009) Using differences among replications of software engineering experiments to gain knowledge. In: 2009 3Rd international symposium on empirical software engineering and measurement, pp 356–366. https://doi.org/10.1109/ESEM.2009.5314236

Larissa Luciano Carvalho M, Lessa gonçalves da Silva M, Gomes G, Santos A, Machado I, Souza ML, Santana de Almeida E (2017) On the implementation of dynamic software product lines: An exploratory study. J Syst Softw 136:74–100. 10.1016/j.jss.2017.11.004

Linsbauer L, Lopez-Herrejon RE, Egyed A (2014) Feature model synthesis with genetic programming. In: Le Goues C, Yoo S (eds) Search-based software engineering. Springer International Publishing, Cham, pp 153–167

Lopez-Herrejon R, Galindo J, Benavides D, Segura S, Egyed A (2012) Reverse engineering feature models with evolutionary algorithms: An exploratory study. pp 168–182. https://doi.org/10.1007/978-3-642-33119-0_13

Lopez-Herrejon RE, Linsbauer L, Galindo JA, Parejo JA, Benavides D, Segura S, Egyed A (2015) An assessment of search-based techniques for reverse engineering feature models. J Syst Softw 103:353–369. https://doi.org/10.1016/j.jss.2014.10.037

Parwananta H, Maghfiroh MFN, Yu VF (2013) Two-phase genetic algorithm for solving the paired single row facility layout problem. Ind Eng Manag Syst 12:181–189

Phadke MS (1989) Quality engineering using robust design. Prentice Hall

Roy R (2010) A primer on the taguchi method society of manufacturing engineers

Sadeghi J, Niaki STA (2015) Two parameter tuned multi-objective evolutionary algorithms for a bi-objective vendor managed inventory model with trapezoidal fuzzy demand. Appl Soft Comput 30:567–576

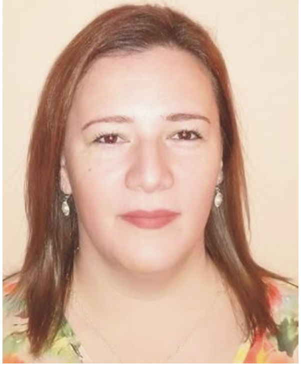
Segura S, Galindo JA, Benavides D, Parejo JA, Ruiz-Cortés A (2012) Betty: Benchmarking and testing on the automated analysis of feature models. In: Proceedings of the Sixth international workshop on variability modeling of software-intensive systems, VaMoS '12. Association for Computing Machinery, New York, pp 63–71. https://doi.org/10.1145/2110147.2110155

She S, Ryssel U, Andersen N, Wasowski A, Czarnecki K (2014) Efficient synthesis of feature models. Inf Softw Technol 56(9):1122–1143. https://doi.org/10.1016/j.infsof.2014.01.012

Shepperd M, Ajienka N, Counsell S (2018) The role and value of replication in empirical software engineering results. Inf Softw Technol 99:120–132. https://doi.org/10.1016/j.infsof.2018.01.006

Yin RK (2008) Case study research : design and methods. SAGE

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Andreea Vescan is an Associate Professor Babeș-Bolyai University, Romania. She obtained the Ph.D. degree in Computer Science with cum laude distinction in 2009 from Babeș-Bolyai University under the supervision of Prof. Militon Frentiu. The research and teaching interests of Dr. Vescan are primarily in formal models for component-based systems, and verification and validation of software systems.

Adrian Pintea is a Master student at Babeș-Bolyai University, Romania. He graduated B.Sc. program in Computer Science at Aurel Vlaicu University in Arad, Romania. The research topic for the bachelor's thesis was the implementation of a 3d game in Unity (game engine). His main research interests are in Software Product Lines and Configurable Systems.

Lukas Linsbauer is currently a postdoctoral researcher at the Institute of Software Engineering and Automotive Informatics at the Technische Universität Braunschweig in Germany. He received his Doctorate in 2016 from the Institute for Software Systems Engineering at the Johannes Kepler University Linz in Austria under the supervision of Prof. Alexander Egyed and Dr. Roberto Erick Lopez-Herrejon. His research interests include highly variable and configurable systems, software product lines, feature-oriented software and systems development, traceability, and version control systems.

Alexander Egyed heads the Institute for Software Systems Engineering (ISSE) at the Johannes Kepler University, Austria. He received his Doctorate from the University of Southern California, USA and worked many years in industry before joining academia. Dr. Egyed was recognized among the Top 10 scholars in software engineering and his work has received numerous awards.

Affiliations

Andreea Vescan[1] · **Adrian Pintea**[1] · **Lukas Linsbauer**[2] · **Alexander Egyed**[3]

Andreea Vescan
avescan@cs.ubbcluj.ro

Adrian Pintea
adrian.pintea1@stud.ubbcluj.ro

Lukas Linsbauer
l.linsbauer@tu-braunschweig.de

[1] Computer Science Department, Babeş - Bolyai University, Cluj-Napoca, Romania

[2] Institute of Software Engineering and Automotive Informatics, Technische Universitat Braunschweig, Braunschweig, Germany

[3] Institute for Software Systems Engineering, Johannes Kepler University, Linz, Austria