Towards a Semiautomatic Tool to Support the Integration of Feature Models

Vinicius Bischoff
Graduate Program in Applied Computing (PPGCA),
University of Vale do Rio dos Sinos (UNISINOS)
viniciusbischof@edu.unisinos.br

Kleinner Farias
Graduate Program on Applied Computing (PPGCA),
University of Vale do Rio dos Sinos (UNISINOS)
kleinnerfarias@unisinos.br

Lucian José Gonçales
Graduate Program on Applied Computing (PPGCA),
University of Vale do Rio dos Sinos (UNISINOS)
lucianj@edu.unisinos.br

Jorge L. V. Barbosa
Graduate Program on Applied Computing (PPGCA),
University of Vale do Rio dos Sinos (UNISINOS)
jbarbosa@unisinos.br

ABSTRACT

The integration of feature models plays a key role in many software engineering tasks, e.g., adding new features to software product lines (SPL) of information systems. Previous empirical studies have revealed that integrating design models is still considered a time-consuming and error-prone task. Unfortunately, integration approaches with tool support are still severely lacking. Even worse, little is known about the effort invested by developers to integrate models manually, and how correct the integrated models are. This paper proposes FMIT, which is a semiautomatic tool to support the integration of feature models. It comes up with a strategy-based approach to reduce the effort that developers invest to combine feature models and increase the amount of correctly integrated models. A controlled experiment was run with 10 volunteers through six realistic integration scenarios. Our results, supported by statistical tests, show that our semiautomatic approach not only reduced the integration effort by 73.01%, but also increased the number of correctly integrated feature models by 43.01%, compared with the manual approach. Our main contributions are a semiautomatic, strategy-based approach with tool support, and empirical evidence on its benefits. Our encouraging results open the way for the development of new heuristics and tools to support developers during the evolution of feature models.

CCS CONCEPTS

• **Software and its engineering → Model-driven software engineering**; • **Integration** → *Empirical studies.*

KEYWORDS

Feature Model, Software Engineering , Controlled Experiment

ACM Reference Format:
Vinicius Bischoff, Kleinner Farias, Lucian José Gonçales, and Jorge L. V. Barbosa. 2019. Towards a Semiautomatic Tool to Support the Integration of Feature Models. In *XV Brazilian Symposium on Information Systems (SBSI'19), May 20–24, 2019, Aracaju, Brazil.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3330204.3330249

1 INTRODUCTION

Software Product Lines (SPL) are widely used approach for the portfolio development of software products [14]. SPL help organizations develop their products from reusable core assets rather than from scratch [12]. Feature models (FM) are considered a prerequisite for software analysis in SPL [5, 13]. A feature can be briefly defined as a functionality (or behavioral) that a software product should provide [9]. FMs can be then seen as a "big picture" of the functionalities of a software system [4]. The adoption of FMs emerges as a result of the internationalization of production processes, becoming common in software-development projects in the industry. Researchers and practitioners have widely used FMs for different purposes, such as: (1) specifying features and their dependencies for automatically deriving products from SPL [7]; (2) describing variability in SPL for aiding the derivation of different products of the line [18]; (3) documenting the features and their valid combinations to enable the strategic reuse of their artifacts [5]; or (4) even assisting developers in integration of features of a family of software systems [2, 7, 8].

Considering that FMs can be created in collaboration by different software-development teams [1, 11], at some point, the FMs created in parallel must be integrated to form an overview of the SPL variabilities. In this sense, the integration of feature models plays a key role in software-development tasks. However, most modeling users find it difficult to apply it. This difficulty comes from the imprecise understanding of functional models and the lack of practical guidelines for modeling features [7, 10].

The integration of feature models can be briefly defined as a set of activities that must be performed on two input models, FM_A and FM_B, to produce a desired output feature model, the FM_{AB}. However, developers may end up not producing the FM_{AB}. Instead, developers typically produce a composed output model, FM_{CM}, with problems (i.e., $FM_{CM} \neq FM_{AB}$) [11]. This is because developers generally cannot properly detect and resolve integration issues, such as conflicts and inconsistencies, because of the problem in question. Therefore, to produce the FM_{AB}, they must invest some effort to resolve such conflicts and inconsistencies in the FM_{CM}. However, if this problem is not resolved properly, inconsistencies will be inserted into the composite feature model. Figure 1 provides an overview of how FM integration can be mediated. We determine three steps: (1) the effort produced by the developers in generating the input models *eff*(FM$_A$, FM$_B$); (2) the effort applied to integrate the models and solve the conflicts, *iff*(FM$_{CM}$); and finally, (3) the effort applied to produce the desired model *diff*(FM$_{AB}$).

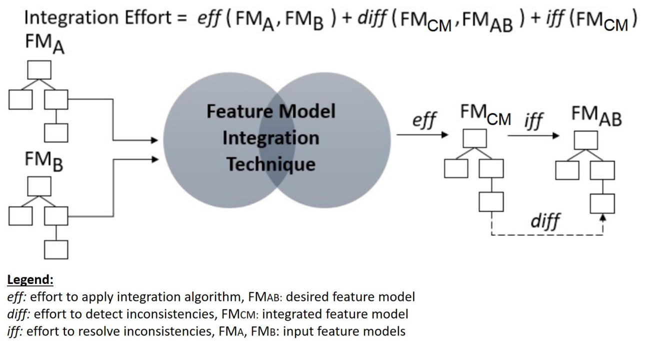

Integration Effort = eff (FMA, FMB) + diff (FMCM, FMAB) + iff (FMCM)
FMA
FMB
Feature Model Integration Technique
eff
FMCM
iff
FMAB
diff
Legend:
eff: effort to apply integration algorithm, FMAB: desired feature model
diff: effort to detect inconsistencies, FMCM: integrated feature model
iff: effort to resolve inconsistencies, FMA, FMB: input feature models

Figure 1: Feature model integration effort producted by the developers.

Previous works [11] have already investigated the effects of compositional tasks on the developers effort and their experiences. They have also revealed that integrating design models is still considered a time-consuming and error-prone task. Farias et al. [11] evaluated the effort invested to compose UML models using techniques based on specification and heuristics. Unfortunately, the integration of feature models has not been enough explored yet. In fact, integration approaches with tool support are still severely lacking [2]. Even worse, little is known about the effort invested by developers to integrate models manually, and how correct the integrated models are.

To account for this, this paper proposes Feature Model Integration Tool - FMIT, which is a semiautomatic tool to support the integration of feature models. It comes up with a strategy-based approach to reduce the effort that developers invest to combine feature models and increase the amount of correctly integrated models. A controlled experiment was carried on with 10 volunteers through six realistic integration scenarios. The volunteers integrated feature models manually, and then they integrate feature models using the FMIT. Our empirical study provides some practical findings about the benefits of the FMIT in relation to the manual integration of feature models.

Our results, supported by statistical tests, show that FMIT semiautomatic approach not only reduced the integration effort by 73.01%, but also increased the number of correctly integrated feature models by 43.01%, compared to the manual approach. The main contributions of this article are a semiautomatic, strategy-based approach with tool support, and empirical evidence on its benefits. The results are encouraging and open the way for the development of new heuristics and tools to support developers during the evolution of feature models.

The remainder of this article is organized as follows. Section 2 describes the related works. Section 3 presents the proposed approach. Section 4 presents the study methodology. Section 5 presents and discusses the obtained results regarding the formulated research questions. Section 6 discusses some threats to validity of our results. Finally, Section 7 presents some conclusions and future works.

2 RELATED WORKS

This section provides an overview of the related works. For this, we have surveyed the current literature for finding studies related to the problematic explored by this paper. Section 2.1 analyzes some related works. Section 2.2 introduces, in turn, a comparative analysis of the analyzed related works.

2.1 Analysis of the related works

We use Google Scholar[1] to select similar works. In total, 4 articles were selected, which are described and analyzed below.

Acher et al. [2]. It introduced a set of operators for the composition between two FMs producing a new model. The proposed operators support inclusion, change, integration and extension between the composition of two models. Each composition is defined by a set of rules that describe the structure resulting from the integration between the models. However, the proposed approach was not evaluated by potential users, disregarding the provision of data related to perceived ease and use, attitudes towards use and the degree of acceptance as a whole.

Benavides et al. [5]. This paper reported an analysis of automation in feature models. They investigate the operations, techniques and tools resulting from the literature outlining the major advances. The studies show a catalog identifying more than 30 operations in the literature and classifying the existing proposals. They evaluated and compared different approaches to automating existing feature models. This work does not focus on the expression of the integration relationship and does not conduct empirical studies with potential users on this topic.

Farias et al. [11]. It conducted an evaluation over the evolution of UML diagrams to add new features or reconciliation models developed in parallel by different software-development teams. The main results suggest that heuristics techniques require less effort to produce the desired model. This article, therefore, reports a controlled experiment that investigates the effort to apply techniques of composition of models and to detect and to solve inconsistencies in the compound models. The work does not focus on the modeling of feature diagrams.

Accioly et al. [1]. This study conducted an empirical study in collaborative environments. The main problem pointed out by the authors refers to the conflicts that arise during the integration of source codes. They point out that different awareness tools have been proposed to alert developers. However, there is not much empirical evidence to support the strategies used by these tools. The article analyzes the effectiveness of two types of code changes as predictors of conflicts. Unlike our study, the study seeks to identify conflicts in source code, as well as was not evaluated by potential users.

Researchers and professionals of industry are looking for solutions that aim to reduce production time and increase the correctness of their models. Tools and strategies use different approaches to both decrease integration effort and improve correctness during

[1]https://scholar.google.com

integration tasks. Hence, developers and researchers end up not knowing which and how the integrate between two input feature models should be performed in practice. Throughout this work, we search analyzes the benefits to facilitate the day to day of the developers, with a view to filling the gaps identified. The following section highlights some gaps by presenting a comparative analysis of the previously discussed studies.

2.2 Comparative analysis of the works

This section contrasts the proposed approach with the previously analyzed studies. This comparison, based on comparison criteria (C), serves to identify some similarities and differences. The comparison criteria are presented below: **Main contribution (C1)**: Studies that have as main contribution ion the use of feature models to express integration relationships or to address research topics related to composition relationships, including conflict representation, conflict detection, and more. **Proposed approach (C2)**: Studies that introduce a new approach that deal with research topics concerning integration, comparison, merge, and/or transformation of models or source codes. **Experimental study (C3)**: Studies that evaluated the proposed approach through empirical studies, including case study, controlled experiment, quasi-experiment or survey. **Context (C4)**: Studies that have been performed with industry professionals or used real-world artifacts in academia. **Participant profile (C5)**: Studies that collected data of the participants to screen and characterize their profile. **Study variables (C6)**: Studies that analyzed the effort to merge models, the correctness of merge relationships and the acceptance of the proposed approach.

Table 1 presents the comparison considering these criteria. It is observed that, only the proposed work fully meets the defined criteria, highlighting the contribution and the differential of this work.

Table 1: Comparative analysis of related works.

Related Work	Comparison Criteria					
	C1	C2	C3	C4	C5	C6
FMIT	●	●	●	●	●	●
Accioly et al. [1]	○	●	●	○	○	○
Acher et al. [2]	●	●	○	○	○	○
Benavides et al. [5]	◐	○	●	◐	○	◐
Farias et al. [11]	○	●	●	●	●	●

Legend
● Meets Fully ○ Does not meet
◐ Meets partially

We present the FMIT in Section 3 and evaluate it in Section 4 by comparing it with a manual approach using the FeatureIDE[2].

3 THE PROPOSED APPROACH

This section presents FMIT, which is a semiautomatic tool to support the integration of feature models. It provides a strategy-based approach to aid developers during the integration of feature models. The FMIT consists of an approach that partially automates the integration of FMs as some types of conflicts like semantic conflict that cannot be automatically resolved [15].

Functionality. Figure 2 shows the feature model of the proposed tool. We note that the first level comprises five features, all of which are mandatory:

Analyze: It is responsible for validating the input and output models. *Compare*: If the feature models to be integrated are valid, then the next step is to compare them. This functionality defines the semantic and syntactic equivalence between the models. *Integrate*: It aims at combining the models based on the equivalence description produced by the compare feature. *Evaluate*: This functionality allows developers to check well-formedness rules of the feature models, but it is not completely implemented. *Persist*: It aims at persisting the models being manipulated in text format (e.g., XML).

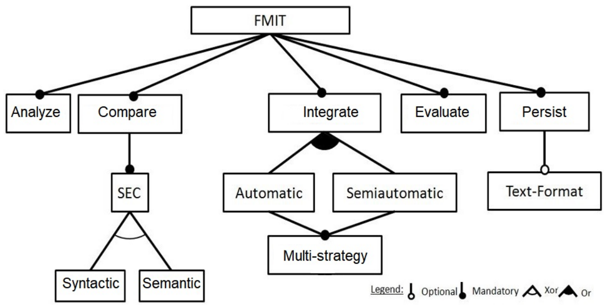


Figure 2: The functionality of the FMIT represented as a feature model.

Interface. Figure 3 illustrates the FMIT'tool interface. This interface is composed by three main sections: (A) Package Explorer, (B) Input Feature Model's, and (C) Console. These sections are described below.

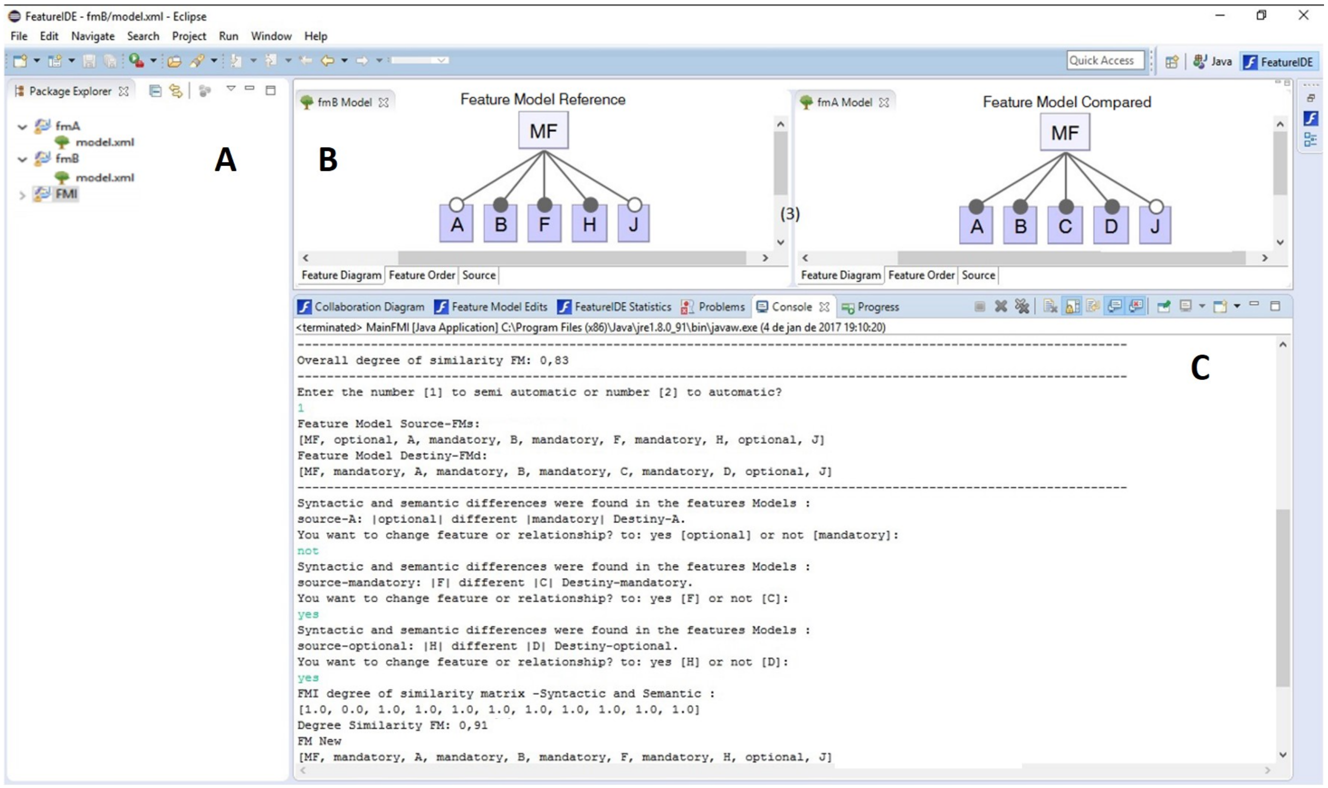
(A) *Package explorer*: It provides the navigation and structure of input features models. The developer browses through these files for information about the project being worked on. (B) *Input feature model*: It presents two input features, FM_A and FM_B. These models are changed and evolved by developers. (C) *Console*: It exhibits the progress of the input feature models. During this progress, developers resolve emerging conflicts by answering questions generated by FMIT at points where the technique cannot decide.

Main characteristics. FMIT presents the following particularities: (1) extensive integration with the Eclipse platform; (2) feature model editor inherited from FeatureIDE tool; (3) settings and constraints editor; (4) statistical data; and, finally, (5) allows integration with software-developed in Java and XML support, as tools that support the modeling of features (for example, AHEAD, FeatureHOUSE, among others).

4 STUDY METHODOLOGY

This section describes the study methodology used to evaluate the FMIT. Section 4.1 presents the objective and research questions that are explored in our evaluation. Section 4.2 formulates the study hypotheses. Section 4.3 discusses details about the study variables and their quantification method. Finally, Section 4.4 shows the adopted experimental design to run the controlled experiment.

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



```
<terminated> MainFMI [Java Application] C:\Program Files (x86)\Java\jre1.8.0_91\bin\javaw.exe (4 de jan de 2017 19:10:20)
------------------------------------------------------------------------------------------------
Overall degree of similarity FM: 0,83
------------------------------------------------------------------------------------------------
Enter the number [1] to semi automatic or number [2] to automatic?
1
Feature Model Source-FMs:
[MF, optional, A, mandatory, B, mandatory, F, mandatory, H, optional, J]
Feature Model Destiny-FMd:
[MF, mandatory, A, mandatory, B, mandatory, C, mandatory, D, optional, J]
------------------------------------------------------------------------------------------------
Syntactic and semantic differences were found in the features Models :
source-A: |optional| different |mandatory| Destiny-A.
You want to change feature or relationship? to: yes [optional] or not [mandatory]:
not
Syntactic and semantic differences were found in the features Models :
source-mandatory: |F| different |C| Destiny-mandatory.
You want to change feature or relationship? to: yes [F] or not [C]:
yes
Syntactic and semantic differences were found in the features Models :
source-optional: |H| different |D| Destiny-optional.
You want to change feature or relationship? to: yes [H] or not [D]:
yes
FMI degree of similarity matrix -Syntactic and Semantic :
[1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
Degree Similarity FM: 0,91
FM New
[MF, mandatory, A, mandatory, B, mandatory, F, mandatory, H, optional, J]
```

Figure 3: An illustrative example of FMIT interface.

4.1 Objective and Research Questions

This study aims to evaluate the impact of the FMIT on the effort invested by participants and the correctness of the integrations. Our participants used the FMIT and the FeatureIDE tool to integrate feature models through realistic scenarios. The objective of this study is organized following the Goal Question Method (GQM) template [19].

Analyze *the integration techniques*
for the purpose of *investigating effects*
with respect to *correctness and effort*
from the perspective of *developers*
in the context of *evolution of feature models.*

The more developers manipulate the models to detect and solve problems, the greater the effort invested [6, 16]. We are concerned with whether FMIT can improve model integration in some way by reducing effort and increasing the number of desired models produced. The explored variables are, in turn, the effort invested by our study participants and the correctness of the integrated models. In this way, the research questions (RQ) are defined as follows:

- **RQ1:** What is the effort invested by developers to integrate feature models?
- **RQ2:** What is the implication of traditional techniques in relation to the production of the correct model?

4.2 Hypothesis Formulation

This section formulates the hypotheses that aim to guide the controlled experiment. In this paper, two hypotheses were formulated. Hypothesis 1 deals specifically with analyzing the effort, while Hypothesis 2 explores the correctness.

Hypothesis 1. The integration technique of the FMs tends at first to be user-friendly due to its simplicity. However, as the possible features and combinations expand, they can become costly as they involve procedures for analyzing the FM$_A$ and FM$_B$ input models to identify possible equivalence between elements of all models. Development teams can invest a lot of effort to integrate models, and these efforts are often not converted into the desired model, i.e., it presents an inconsistent model [11, 15]. Therefore, the first hypothesis evaluates whether the semiautomatic integration technique, based on the proposed prototype, reduces the integration effort, helping the analysts and developers to produce the desired models consuming less time. Based on this conjecture, the null and alternative hypotheses on the Integration Effort (IE) are presented below:

Null Hypothesis 1 (H_{1-0}): The proposed semiautomatic technique does not reduce the integration effort by producing the desired model, FM$_{AB}$, from FM$_A$ and FM$_B$.
$\boldsymbol{H_{1-0}}$: Effort$(FM_A, FM_B)_{Semiautomatic} \geq$ Effort$(FM_A, FM_B)_{Manual}$

Alternative Hypothesis 1 (H_{1-1}): The proposed semiautomatic technique reduces the integration effort by producing the desired model, FM_{AB}, from FM_A and FM_B.
***H*₁₋₁:** $\text{Effort}(FM_A, FM_B)_{Semiautomatic} < \text{Effort}(FM_A, FM_B)_{Manual}$

In analyzing the first hypothesis, we intend to evaluate whether the proposed technique reduces the integration effort to produce the desired model, FM_{AB}, thus generating empirical evidence on how the proposed techniques accommodate changes into the FMa from the FM_B to produce the FM_{AB}.

Hypothesis 2. The inconsistencies affect the composition of models due to conflicting changes, affecting the syntactic and semantic properties of the model, and their integration does not coincide with the desired model [1, 11]. The resolution of problems found in models is influenced by the presence or not of inconsistencies in the integrated output model, so it is necessary to evaluate the correctness of the changes made. This hypothesis seeks to assess whether the integration of input models performed so semiautomatically assists developers in models reduction produced with inconsistency. That is, models that do not have errors in comparison to the desired model. Based on this statement, we define the null and alternative hypothesis comparing the Correct Integration (CI) presented below:

Null Hypothesis 2 (H_{2-0}): The proposed technique does not significantly favor the production of the desired model, FM_{AB}, from FM_A and FM_B.
H_{2-0} : $\text{Correctness}(FM_A, FM_B)_{Semiautomatic} \leq \text{Correctness}(FM_A, FM_B)_{Manual}$

Alternative Hypothesis 2 (H_{2-1}): The proposed technique significantly favors the production of the desired model, FM_{AB}, from FM_A and FM_B.
H_{2-1} : $\text{Correctness}(FM_A, FM_B)_{Semiautomatic} > \text{Correctness}(FM_A, FM_B)_{Manual}$

When analyzing the second hypothesis, we want to verify the correctness rate of the integrations performed by the participants in this experiment. With this, we intend to identify the inherent conflicts of failures produced during this activity. To run the analysis of the above hypothesis some variables were defined. These variables are detailed in the next section.

4.3 Study Variables

The independent variable is the use of integration technique: semiautomatic and manual. We control the use of the FMIT (automatic) and FeatureIDE (manual) for understanding their impact on the dependent variables: effort and correctness. The effort is quantified based on the minutes invested to combine two feature models. The correctness refers to the number of correctly integrated by amount the integration realized. Table 2 shows the summary of the explored variables in this study.

4.4 Experiment Process

Figure 4 shows the adopted experimental process. This process follows empirical guidelines reported in [19], and has been validated in previous empirical studies [11]. This experimental process is composed of three phases which are described below:

Table 2: Description of the dependent and independent variables.

Type	Variable	Scale
Independent variables	Main	Nominal: Integration techniques
Dependent Variables	Correctness Effort	Ordinal: 0 or 1 Interval [0..60]

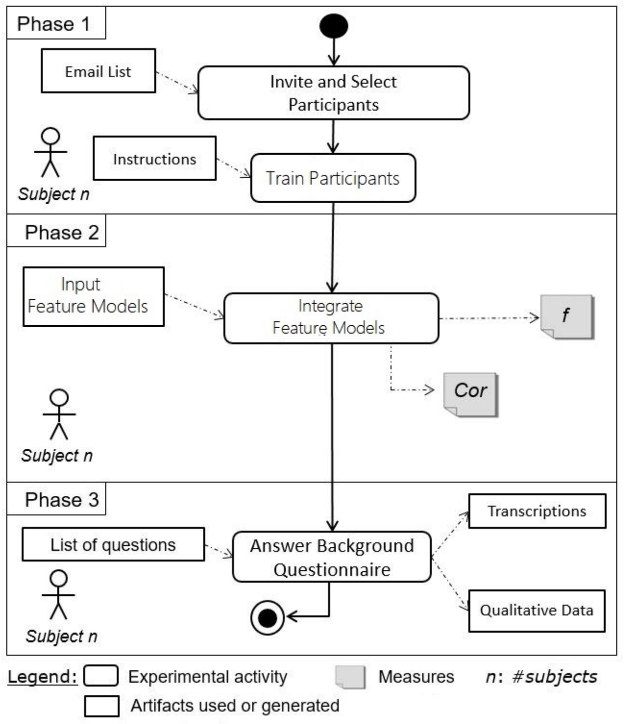


Figure 4: The experimental process used to evaluate the FMIT.

First phase: Selection and training of participants. The first activity, invite and select the participants, concerned on recruiting the volunteers to execute the experiment. For this, we elaborated a list of possible candidates and then generated an e-mail list. These candidates in this list were invited. Next, in the training activity, all participants that accepted to execute the experiment were trained. Thus, we explained the context of the experiment, and the steps of the experiment based on a list of instructions. This preparation ensured that they acquired the necessary familiarity with the activity.

Second phase: Experiment execution. The focus of this phase was to realize the integration itself. So, the selected participants used two tools, FMIT (semiautomatic), with integration support and FeatureIDE (manual), without integration support through six integration scenarios. Each integration scenario is equivalent to one question (experimental task). Each participant performed a total of six experimental tasks. The models used in this experiment were small diagrams of features, with about 10 features, 7 relationships, 3 depth levels, and 3 average conflicts per model presented. The model size needs to be small due to time constraint, and the large models might also influence the participants during the execution of the tasks. The data related to the correctness (Cor) and integration effort (f) were collected and then manipulated.

Third phase: Participant background. Finally, in the last activity, Respond Background Questionnaire, we distributed to the participants, a list of questions asking information about their background, such as their respective teach institutions, years of experience with software modeling and development. This information is important to check the homogeneity of the sample. In addition, they gave opinions about the experiment. To sum up, they fulfilled a questionnaire about their knowledge backgrounds, and skills. The output of this activity is the transcriptions of this interview, and qualitative data about their background.

5 RESULTS

This section details the results of the performed experiment. Section 5.1 presents the analysis procedures used to explore the collected data. Section 5.2 introduces the results of the first hypothesis, that investigates the integration effort. Section 5.3 shows the results of the second hypothesis that investigates the correctness of the feature models.

5.1 Analysis of collected data

To analyze the normality of the samples, we chose to perform two tests: *Kolmogorov-Smirnov*, used for large samples ($n \geq 30$) and *Shapiro-Wilk* reduced ($n < 30$) [17]. Table 3 presents the result of the descriptive level, the Kolmogorov-Smirnov statistic, with a significance level of *Lilliefors* for normality test, and Shapiro-Wilk, the results returned from significance, also known as p-value < 0.05 for both tests in the analyzed samples. Thus, it can be affirmed that a level of significance of 5% does not come from a normal sample.

Table 3: The results of the normality tests.

	Kolmogorov-Smirnov a			Shapiro-Wilk		
Effort	**Statistic**	**DF**	**Sig.**	**Statistic**	**df**	**Sig.**
Semiautomatic	0.206	30	0.002	0.881	30	0.003
Manual	1.219	30	0.001	0.814	30	0.000
Correct	**Statistic**	**DF**	**Sig.**	**Statistic**	**df**	**Sig.**
Semiautomatic	0.354	30	0.000	0.637	30	0.000
Manual	0.537	30	0.000	0.275	30	0.000

Legend: a* - Correlation of Significance of Lillieforrs

Kolmogorov-Smirnov and *Shapiro-Wilk* [11, 17] tests of normality indicated that the our data were not normally distributed, so non-parametric tests should be applied. Therefore, the *Wilcoxon* test was applied to analyze the effort, and the *McNemar* test was used to the correctness. These methods were used to test the formulated hypotheses (Section 4.2). We test the formulated hypotheses in the following sections.

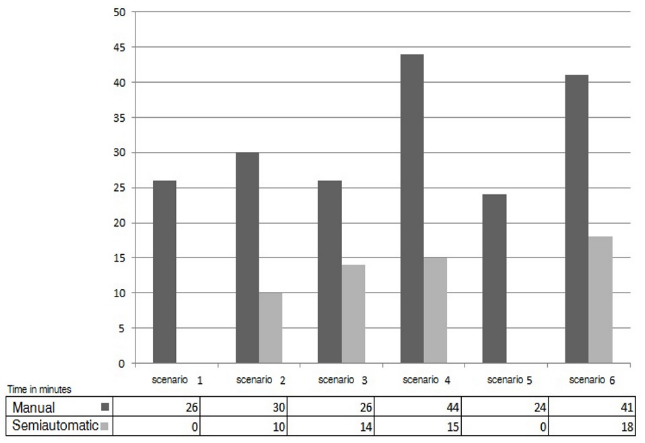
5.2 RQ1: Effort and Integration

Descriptive statistics. Table 4 presents the computed descriptive statistics. The results indicates that the use of FMIT helped to reduce the integration effort. On average, the effort reduced 70.13% using the FMIT, compared with the manual integration using the FeatureIDE. In addition, we noticed that the standard deviation among the techniques investigated did not present data dispersion.

The *Wilcoxon* test is a non-parametric static hypothesis test applied to compare the mean of two related samples (i.e., a paired difference test). It can be used as an alternative to *t-test*. The *t-test* is considered a hypothesis test that uses statistical concepts to reject or not a hypothesis, and is applied to samples with normal distribution. [3].

Hypothesis Testing. Table 4 shows the collected data related to the formulated hypotheses. Considering effort variable, the obtained p-value is lower that 0.05. This means that the null hypothesis H_{1-0} can be rejected. The p-value is 0.001, with a confidence interval of 95 % in a sample of 30 pairs. The non-parametric *wilcoxon* test which can be seen in Table 4.

Figure 5 shows the comparative in minutes of each scenario. In this way, we obtain the total integration effort. The time in minutes is given by the sum of the questions answered by the participants. Scenario 6 in both questions were the one that presented greater effort to solve the existing conflicts. Integration scenarios 1 and 5 did not present conflicts. We can say that the participants invested less effort to perform integration using the semiautomatic technique in relation to the manual technique.



	scenario 1	scenario 2	scenario 3	scenario 4	scenario 5	scenario 6
Manual ■	26	30	26	44	24	41
Semiautomatic ■	0	10	14	15	0	18

Figure 5: Effort applied in relation to manual technique and FMIT (semiautomatic).

5.3 RQ2: Correctness and Integration

Descriptive statistics. The data of the descriptive statistics indicate that the average of using the FMIT (semiautomatic) tool was 0.93 and FetureIDE (manual) of 0.53 hits. We can confirm the best performance of the semiautomatic technique. Likewise, a 43.01% increase in model integrations is confirmed. The use of tools that support integration better lead to the level of assertiveness of developers in their tasks.

McNemar test is used to analyze the efficiency of a given technique, that is, it aims to evaluate the efficiency of situations, "before and after", where each sample is used, and the measurement is done at the level of a scale nominal or ordinal. This test is applied to

Table 4: Descriptive statistics related to the effort and correctness.

Variables	Treatment	Min	25th	MD	75th	Max	Avg.	SD	%Diff	W p-value	N p-value
Effort	Semiautomatic	0	0	2	3	5	1.90	1.66	70.13	0.001	-
	Manual	4	5	6	7	12	6.26	2.00			
Correct	Semiautomatic	0	1	1	1	1	0.93	0.25	43.01	-	0.002
	Manual	0	0	1	1	1	0.53	0.50			

Legend: Standard Deviation (SD), Minimum (Min), First Quartile (25th), Median (MD), Third Quartile (75th), Maximum (Max), Average (Avg.), Percentage Difference (% Diff), McNemar test (N), Wilcoxon test (W)

dichotomous variables, that is, to samples that only take two values, for example, 0 and 1 [3, 11].

Hypothesis Testing. Analyzing H_{2-0}, one has to investigate the null hypothesis, the non-parametric *McNemar* test which can be observed in Table 4. It is possible to verify that the collected statistic of the significance is 0.002, with a confidence interval of 95% in a sample of 30 pairs. This value indicates that the second null hypothesis (H_{2-0}) can be rejected. Since the p-value is less than 0.05, one can conclude that there is evidence that the semiautomatic technique is more effective than the manual technique.

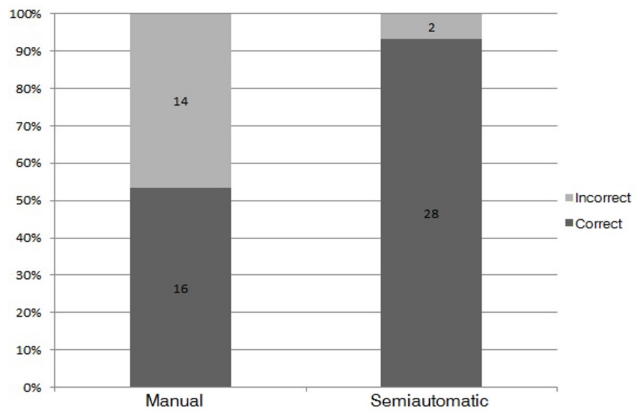


Figure 6: Number of correct responses according to manual and FMIT (semiautomatic) technique

Figure 6 shows the comparison between models produced by semiautomatic and manual techniques. This graphic refers to the correctness of the models, that is, the desired model FM_{AB}. The number of FMs incorrectly integrated by the tool without composition support corresponds to 47% (14/30), and correctly integrated represents 53% (16/30). On the other hand, the use of a tool that supports the composition of models obtained an assertiveness of 93% (28/30), and got 7% (2/30) incorrectly integrated. We observed in this experiment that the consent of the developers is greater when there is a tool indicating the conflicts. We believe that this fact was a facilitator in managing the information produced by the tool during the integration task.

6 TREATS TO VALIDITY

This study presents threats to validity. This section discusses the strategies applied to manage these threats that range from conclusion, construct, and external ones.

Conclusion validity. It considers the ability to obtain the correct conclusion based on the results from the experiment, through the choice of statistical methods, assuming the sample size and the reliability of the measurements [19]. In this criterion the experiment produced a sample of 60 models of features, whose descriptive statistical analysis identified that the obtained data did not adhere to a normal distribution. Thus, we applied non-parametric tests, i.e., the *Wilcoxon* test to measure the effort used in the detection of conflicts, and the *McNemar* test to measure the correctness of the models produced with a confidence interval of 95%.

Experiment validity. It takes into account the relationship between theory and observation, i.e., the results obtained through questionnaires or experiments and are related to the expectations of the studied theory [11, 19]. The experiments carried out in this research were planned to measure the integration effort, as well as to quantify the correctness of the models produced by the participants, being this strategy already adopted in previous studies [11]. In order to meet the execution and correction procedures of the experiment, they were carefully planned, following good practices found in [19].

External validity. It considers the conditions that allow to generalize the collected results to the industrial practice or to the most realistic possible [11, 19]. In this context, we selected the participants who had adequate training to practice the activity related to the experiment, that is, all the participants had a Master's degree in Applied Computing, besides of having experience in the area of software development or modeling. In addition, the tools and equipment used in the experiment are equivalent to the one practiced in industry, and the application of the whole experiment occurred at the University of Vale do Rio dos Sinos.

Given that these criteria may happen in practice, we could conjecture our results may be generalized, at some point, to other contexts having similar settings found in our experiment.

7 CONCLUSION AND FUTURE WORK

This research investigated the influence of the use of tool support by developers to integrate models, mainly to support the resolution of conflicts between parts of the FMs to be integrated. A controlled experiment was carried out to evaluate the understanding of the developers during the integration activities of feature models. The FMs used in the experimental tasks had syntactic and/or semantic conflicts. The developers' understanding was investigated through the analysis of the effort applied to resolve the conflicts, as well as the correctness of the activities performed. Two hypotheses were formulated and tested to analyze the effort and correctness, respectively.

The main findings were that the use of the tool influenced the effort to resolve conflicts, as well as the correctness of conflict resolution. The obtained results indicated that the effort to resolve conflicts reduced by 73%, while the correctness rate increased by 43%. Moreover, this study allowed us to: (1) evaluate the use of semiautomatic integration techniques to assist the developers in carrying out the integration tasks; (2) analyze how conflicting models affect the production of the desired models; and (3) show the importance of empirical studies to generate knowledge related to the practice of integrating feature models. As future work, we will replicate the study using different sizes of feature models and a larger sample of participants to check whether (or not) our findings are confirmed or not.

ACKNOWLEDGMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior -Brasil (CAPES) - Finance Code 001, the Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS), and the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

REFERENCES

[1] Paola Accioly, Paulo Borba, Léuson Silva, and Guilherme Cavalcanti. 2018. Analyzing conflict predictors in open-source Java projects. In *15th International Conference on Mining Software Repositories*. 576–586.

[2] Mathieu Acher, Philippe Collet, Philippe Lahire, and Robert France. 2010. Comparing approaches to implement feature model composition. In *European Conference on Modelling Foundations and Applications*. 3–19.

[3] James Aldrich. 2018. *Using IBM® SPSS® Statistics: An interactive hands-on approach.*

[4] Shahin Ashkiani and Cecilio Molinero. 2017. Visualization of cross-efficiency matrices using multidimensional unfolding. *Recent Applications of Data Envelopment Analysis* 978, 1 (2017), 226.

[5] David Benavides, Sergio Segura, and Antonio Cortés. 2010. Automated analysis of feature models 20 years later: A literature review. *Information Systems* 35, 6 (2010), 615–636.

[6] Vinicius Bischoff, Kleinner Farias, and Lucian Gonçales. 2018. Evaluating the Effort of Integrating Feature Models: A Controlled Experiment. (2018), 202–204.

[7] Vinicius Bischoff, Kleinner Farias, Lucian Gonçales, and Jorge Barbosa. 2018. Integration of feature models: A systematic mapping study. *Information and Software Technology* (2018).

[8] Hugo Bruneliere, Florent Kerchove, Gwendal Daniel, and Jordi Cabot. 2018. Towards scalable model views on heterogeneous model resources. In *21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*. 334–344.

[9] Andreas Classen, Patrick Heymans, and Pierre Schobbens. 2008. What's in a feature: A requirements engineering perspective. In *International Conference on Fundamental Approaches to Software Engineering*. 16–30.

[10] Khanh Dam, Alexander Egyed, Michael Winikoff, Alexander Reder, and Roberto Herrejon. 2016. Consistent merging of model versions. *Journal of Systems and Software* 112 (2016), 137–155.

[11] Kleinner Farias, Alessandro Garcia, Jon Whittle, Christina Chavez, and Carlos Lucena. 2015. Evaluating the effort of composing design models: A controlled experiment. *Software & Systems Modeling* 14, 4 (2015), 1349–1365.

[12] Daniela Lettner, Klaus Eder, Paul Grunbacher, and Herbert Prahofer. 2015. Feature modeling of two large-scale industrial software systems: Experiences and lessons learned. In *18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. 386–395.

[13] Xiaoli Lian, Li Zhang, Jing Jiang, and William Goss. 2018. An approach for optimized feature selection in large-scale software product lines. *Journal of Systems and Software* 137 (2018), 636–651.

[14] Ethan McGee and John McGregor. 2017. A realization effort estimation model for dynamic software product lines. In *21st International Systems and Software Product Line Conference-Volume B*. 111–116.

[15] Gleiph Menezes, Leonardo Murta, Marcio Barros, and Andre Hoek. 2018. On the nature of merge conflicts: A Study of 2,731 Open Source Java Projects Hosted by GitHub. *IEEE Transactions on Software Engineering* (2018).

[16] Anderson Oliveira, Vinicius Bischoff, Lucian Gonçales, Kleinner Farias, and Matheus Segalotto. 2018. BRCode: An interpretive model-driven engineering approach for enterprise applications. *Computers in Industry* 96 (2018), 86–97.

[17] Nornadiah Razali, Yap Wah, et al. 2011. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics* 2, 1 (2011), 21–33.

[18] Sergio Segura, David Benavides, Antonio Cortés, and Pablo Trinidad. 2008. Automated merging of feature models using graph transformations. In *Generative and Transformational Techniques in Software Engineering II*. 489–505.

[19] Claes Wohlin, Per Runeson, Martin Höst, Magnus Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering.*