Experimental Evaluation of FMCheck: A Replication Study

Iuri S. Souza[1], Rafael M. de Mello[2], Eduardo S. de Almeida[1], Cláudia M. L. Werner[2], Guilherme H. Travassos[2]

[1]Departamento de Ciência da Computação, UFBA- Universidade Federal da Bahia

[2]Programa de Engenharia de Sistemas e Computação (PESC), COPPE/UFRJ
Universidade Federal do Rio de Janeiro Caixa Postal 68.511 – 21.9451-970 – Rio de Janeiro – RJ – Brasil

{iurisin,esa}@dcc.ufba.br, {rmaiani, werner, ght}@cos.ufrj.br

Abstract. *Software Product Lines are usually specified using feature models. A hierarchically arranged set of features with different relationships among them represents a feature model. However, there is a lack of techniques to support the detection of semantic defects in feature models. In this context, it was recently developed FMCheck, a checklist-based inspection technique to support the detection of defects in feature models. The results of a first study conducted by FMCheck's developers indicated its feasibility (more effective) when compared to ad-hoc techniques. This paper reports the replication accomplished by an independent research group following a different experimental design but using the same artifacts. The obtained results strengthened the previous findings, indicating that FMCheck is more effective than ad-hoc inspections. However, additional replications should be performed with different experimental designs to understand better the influence of the artifacts inspected over such findings.*

1. Introduction

Software Product Line (SPL) is a key approach to support software reuse. An SPL represents a group of software-intensive systems sharing a common, managed set of *features*, meeting the specific needs of a particular market or mission. Such systems are developed from a common set of core assets in a prescribed way [Northrop 2002]. A feature can also be defined as a prominent user-visible aspect, quality, or characteristic of a software system or systems [Kang et al. 1990]. Based on features, increments in program functionality are established, and different software products can be derived from an SPL [Batory et al. 2006]. SPLs usually use *feature models* in their specifications, represented as a hierarchically arranged set of features with different relationships among them. Feature models capture product line information regarding common and variant features at various levels of abstraction [Benavides et al. 2010], specifying the relationship of each feature with the domain, the dependency relationships among features and corresponding feature interactions constraints.

Although the expected benefits on the use of feature modeling for specifying SPLs, it also introduces a new range of anomalies that could significantly impact the quality of the final software products [de Mello et al., 2014]. Even though there are some approaches for covering the detection of syntax anomalies in feature models

[Benavides et al. 2010], a recent literature review indicated the lack of techniques to support the semantic verification of feature models [de Mello et al. 2014], including the absence of inspection techniques. To fill this gap, de Mello et al. proposed *FMCheck*, an inspection technique for detecting semantic defects in feature models [de Mello et al. 2014]. Such technique was designed to support inspections individually conducted, helping its users on observing whether a given feature model is correctly modeled and best suited to represent a specifically described domain.

Two activities had evaluated FMCheck feasibility. First, a proof of concept was performed with the participation of two members of the Experimental Software Engineering Group at Federal University of Rio de Janeiro (COPPE/UFRJ, Brazil) applying the technique in a specific mobile devices domain [de Mello et al. 2014]. The feedback reported by these participants indicated the completeness of FMCheck and its applicability for supporting the intended activity while the participants reported few false positives. Also, both participants reported a high incidence of same defects. Next, a first experimental study (*in vitro*) was designed involving 14 students (four undergraduate students and ten graduate students) from a Software Reuse course at COPPE/UFRJ. In such study, each participant was invited to perform *ad-hoc* inspections over two distinct domains (first trial) and then perform inspections applying FMCheck over two other distinct domains (second trial). As a result, it was observed that FMCheck was significantly more effective, identifying 51.3% more defects than *ad-hoc* inspections [de Mello et al. 2014]. However, regarding FMCheck efficiency (number of defects/ time), no significant difference was observed. Although the experimental rigor applied for conducting the experimental activities, it is important to highlight the following main threats to validity:

- T1: The same research group that developed the technique also conducted the experiment;
- T2: It used a small and local sample;
- T3: It used only four feature models from four different domains;
- T4: Effectiveness was calculated based on a previously limited set of known defects in the inspected artifacts;

One can see that threats to validity T1, T2 and T3 are commonly observed in Software Engineering (SE) controlled studies in which subjects are individuals. Typically, a research group develops a new technology, and it is also the first one to empirical evaluating the technology (T1). In such evaluations, predominantly restricted groups of individuals are available to compose experimental samples (T2), usually students or research colleagues. As a consequence, the set of different objects used in the evaluation is also reduced (T3). An alternative to overcome the sampling limitation is replicating the experiment and then aggregating the results obtained in both trials. However, due to the nature of software inspections, it is recommended to preserve the same set of objects used and change its distribution to support comparison between results and to provide a more accuracy oracle of known defects, mitigating T4.

Thus, this paper presents a replication of the mentioned *quasi*-experiment, conducted by another research group at Federal University of Bahia (UFBA), Salvador, Brazil. Through such replication, threats T1 and T4 could be mitigated. Although researchers from the Experimental Software Engineering (ESE) Group (COPPE/UFRJ) contributed in the results analysis, characterization data was strategically omitted from them until the conclusion of the data analysis. T2 was partially preserved for each single trial (small samples), but now the results obtained from both small (and different) samples can be aggregated, allowing strengthening the evidence. Also, the experimental design (tasks assignment) was arbitrarily changed in this replication. Thus, the presented replication can be classified as a *changed protocol/ experimenters replication* following the classification proposed by Gómez et al. [Gómez et al. 2014] to replications in SE experiments, as exemplified in [de Mello et al. 2015]. The remainder of this paper is organized as follows: Section 2 presents the related work. Section 3 provides an overview of FMCheck's checklist. Section 4 details the context, the design, and results of the experimental study carried out in this work. Finally, Section 5 describes the conclusions and future directions.

2. Related work

The quality of software reuse artifacts has been a notorious issue and received relative contributions to address this. Next subsections discuss approaches for supporting verification of such artifacts based on three perspectives: *automated feature model checking*, *feature model inspection* and *inspection of textual feature specifications*.

2.1 Automated Feature Model Checking

Some approaches for detecting anomalies in feature models use heuristics based on syntactic and automated model checking. Benavides et al. performed a literature review on papers published about studies proposing automated analysis of feature models from 1990 to 2009 [Benavides et al. 2010]. Based on the analysis of 53 primary studies, the authors reported 30 operations of analysis within four different groups of proposals to automate those operations. Towards automating feature models analysis, Benavides et al. specified a proposal based on a theory of diagnosis to represent the problem of error detection and explanation in general terms. The Feature Model Analyzer tool (FAMA) implemented it by using an abstract solution through Constraint Satisfaction Problems (CSP) solver [Benavides et al. 2007]. The authors also reported an evaluation of the proposed approach during a Software Product Line development project to build a set of Enterprise Resource Planning (ERP) products. Based on the assessment reported the authors claimed that the approach has supported the evolution of the ERP feature model, guaranteed the production of an error-free feature model and reduced the time invested for developing feature models. However, the paper did not present any metrics or dataset supporting such findings.

In recent work, Zhang et al. presented an approach for identifying and validating feature models, supporting the detection of defects based on different relationships among features and their propagation [Zhang et al. 2013]. The approach defines the rules to identify *dead* features and *false* variable features (optional features), sets two algorithms to support automation of the feature model errors detection in a feature model validation tool (FMV-Tool) and provides explanations about the feature model errors for the users. Zhang et al. also reported a comparative evaluation study between FMV-Tool and FAMA [Benavides et al. 2007] tools using feature model examples randomly generated. Based on this evaluation, the authors claimed that FMV-Tool is more efficient than FAMA tool in most cases in which the number of different relationships is not significant. Again, the paper did not present any metrics or dataset to support the reported findings and conclusions. Therefore, one can see the approaches presented in this subsection are concerned with avoiding the incorrect modeling of features and supporting the development of SPLs, which can be useful to syntactic verification in large scale. However, they are unable to support the verification of whether a given feature model is best suited to represent a particular domain (semantic verification), which is typically supported by inspection techniques.

2.2 Feature Model Inspection

Recently, de Mello et al. [de Mello et al. 2014] reported the second trial (updated) from a comprehensive *quasi*-systematic literature review performed to identify verification technologies concerned with SPL in the technical publications [de Mello et al. 2012]. The literature review analyzed 134 papers, full reading pre-selection, aiming to answer the research question: *"What are the existing techniques for inspecting software artifacts developed for reuse?"* The review selected six papers presenting four distinct inspection techniques for software reuse artifacts. However, only FMCheck [de Mello et al. 2012], a checklist-base inspection technique, was identified to support the detection of defects in feature models. The FMCheck technique is composed of the following three main activities:

- *Feature Model Characterization:* in this activity, the domain analyst or the domain designer should fill a model characterization questionnaire. This questionnaire collects the information (such as Domain Engineering Stage and Feature Model Notation) needed to configure the inspection checklist (presented in Section III), to avoid unnecessary verification items for a particular context.
- *Checklist Configuration:* in this step, the inspection moderator selects the checklist verification items to be used in the inspection, aided by a traceability table relating each answer collected by the model characterization questionnaire.
- *Feature Model Inspection:* the customized checklist is then individually applied by one or more reviewers, each one producing his/her discrepancy report describing each defect, its defect category, and location.

An additional work was found out of the systematic review results. Cunha et al. [Cunha et al. 2012] proposed SPLIT (Software Product Line Inspection Techniques), a set of checklist-based techniques for comparing feature models with the product map and for verifying the consistence between such artifacts and the software requirements specification. Different from FMCheck, the SPLIT checklists cannot be tailored based on inspected SPL characteristics. To evaluate SPLIT feasibility the authors compared the amount of defects found by inspectors using SPLIT with another approach over a single domain with only 13 features. The evaluation pointed out that inspections supported by SPLIT found greater number of defects than the other approach.

2.3 Inspection of Textual Feature Specifications

In addition to the feature models, textual feature specifications could also be supported by inspection techniques. Souza et al. [Souza et al. 2013] performed an empirical study to understand how inspection should be suited on textual feature specifications in the product line context. They investigated the effects of applying a checklist based inspection approach in textual features. The checklist was composed of questions such as "*Is the textual feature specification enough to model the domain graphical feature model?*" and "*Is there any conflict of priority (e.g., mandatory feature requesting* *optional feature) or dependency (e.g., mutually exclusive features) among features that have a relationship?*". The dataset was gathered from an industrial SPL project for reengineering medical and health information systems. The study sample was analyzed using statistical and economical techniques, which showed that incompleteness and ambiguity reported higher non-conformity occurrences and optional features presented a higher non-conformity density than mandatory features.

3. FMCheck's Checklist Overview

The main contribution of this work is to present a replication of the first experimental evaluation of FMCheck, an inspection technique to support the identification of defects in feature models. De Mello et al. [de Mello et al. 2014] depicted the FMCheck verification items from a summary of 48 discrepancy cases identified from examples analyzed using FODA [Kang et al. 1990] and other feature modeling notations. Such discrepancy cases are fundamentally related to *consistency*, *clearness*, *correctness*, *relevance* and *completeness* of a feature model in comparison to its corresponding domain textual description. The defects identified by FMCheck are classified into five types, according to the following categorization [Shull et al. 2000], [Rocha et al. 2001]:

- *Omission:* Some information from the domain was not properly included in the feature model.
- *Incorrect fact:* Some information or behavior from the feature model contradicts its domain specification.
- *Ambiguity:* Some Information from the feature model is not clear, allowing multiple interpretations for the specified domain.
- *Inconsistency:* Some feature model element is not consistent with another element from the same feature model.
- *Extraneous information:* Some Information in the feature model is outside the domain scope.

The FMCheck checklist is composed of 34 verification items (questions) distributed into three verification groups: *individual verification of each feature*, *verification of relationships between features*, and *verification of composition rules*. Following subsections briefly present the verification groups that compose the checklist, thoroughly described at [de Mello et al. 2014]. Besides, examples of defects reported by subjects on inspecting the *hospitality* domain in the context of the study presented in Section 4 are also presented.

3.1 Individual Verification of each Feature

This group aims to ensure that each feature has been described correctly, clearly and objectively. The verification items of this group (exemplified in Table 1) also support checking if each feature belongs to the modeled domain. Figure 1 shows an excerpt from the feature model describing the *hospitality domain* using the Odyssey-FEX notation [Blois et at. 2006]. One can see the feature "*Booking Confirmation*" was modeled as a *conceptual feature*. However, the textual description of the domain indicates that "*Booking Confirmation*" is a *functional feature*. Thus, this occurrence is a defect (*Incorrect Fact*) which detection could be supported through the verification item *1* (Table 1). Besides, since there is no mention of "*Connection with Card Operator*" and "*Connection with Bank Operator*" features in the domain description, they are

Extraneous Information introduced by the modelers. These defects could be detected through the verification item *11* (Table 1).

Table 1. Excerpt of verification items for individual verification of each feature.

Id.	Description
1	Are all the features clearly and correctly described?
2	Is the described optionality of each feature (optional/mandatory classification) by the domain specification?
3	Is it possible to identify the feature category by its description on the domain?
11	Is there some feature in the model that, although correct, is out of the domain scope?
12	Are there different features in the model that represent the same domain concept?
13	Is there any domain concept that has been omitted from the model?

3.2 Verification of Relationships between Features

The verification items of this group aim to verify how the representation of the relations between features renders the model understandable, deployable, and compliant with the domain. Table 2 shows an excerpt of the verification items of this group. Through the excerpt of the hospitality feature model presented in Figure 1, one can see an implementation relationship between the features *"SSL"* and *"Card Purchase Authorization*." However, since it was not identified any relationship between such features in the domain textual specification, the verification item 17 (Table 2) could be used to determine such *inconsistency*.

Table 2. Excerpt of verification items for the relationship between features.

Id.	Description
14	Are the variabilities of the domain adequately represented as groups of alternatives (variation point and its variants)?
15	Are the cardinalities of the variation points correct?
16	Are the variation points clearly described, reflecting the meaning of their variants?
17	Are there two or more features having a relationship in the model without defining this relationship in the domain?
18	Is there some relationship described in the domain that has not been informed in the model?
19	Is the established hierarchy between each feature compliant with the domain?
25	Is there any feature in the model contradicting other features?
26	Does the root feature help to understand the meaning of the domain?
27	From a general perspective, is it possible to understand the domain from the features represented in the model?
28	Does the model describe the domain in an appropriate level of detail to be understood from the intended perspective?
29	Does the model have the sufficient features to guide the domain implementation?

3.3 Verification of Composition Rules

The five verification items presented in Table 3 guide the inspector in checking the clearness, completeness, correctness, relevance, and consistency of the feature model composition rules as established in FODA notation [Kang et al. 1990]. For instance, after analyzing the hospitality domain description, composition restrictions regarding the relationship between the features "Hospitality" (root feature) and "Booking Confirmation" not represented in the feature model were identified. The detection of these omissions is supported by the verification item 33.

Table 3. Excerpt of verification items for composition rules.

Id.	Description
30	Are all the composition rules clearly and objectively described, being in compliance with the domain description?
31	Is there any composition rule that contradicts another one in the same model?
32	Is there any composition rule that is not applied to this domain, although it is correct?
33	Are all domain composition rules adequately represented in the model?
34	Does the model present sufficient composition rules to guide its implementation?

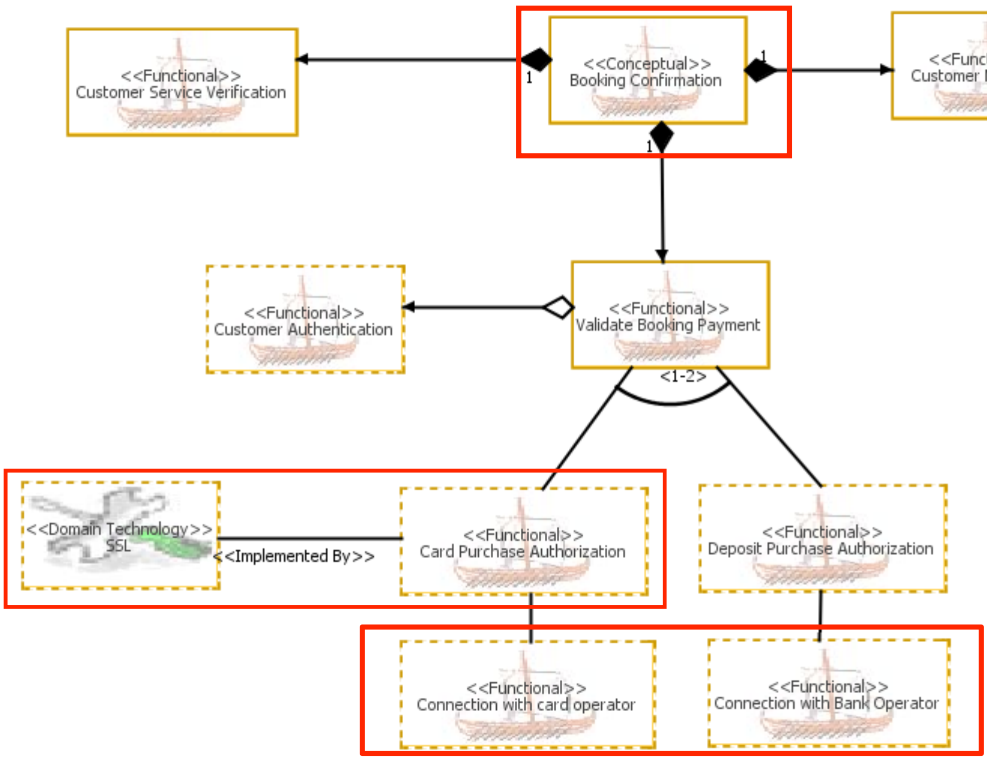
<<Functional>> Customer Service Verification
<<Conceptual>> Booking Confirmation
<<Functional>> Customer Notification
1
1
1
<<Functional>> Customer Authentication
<<Functional>> Validate Booking Payment
<1-2>
<<Domain Technology>> SSL
<<Implemented By>>
<<Functional>> Card Purchase Authorization
<<Functional>> Deposit Purchase Authorization
<<Functional>> Connection with card operator
<<Functional>> Connection with Bank Operator

Figure 1. Excerpt of the Hospitality Feature Model represented through Odyssey-FEX notation, highlighting the location of the defects reported.

4. The Study Replication

We conducted an operational replication of the *quasi*-experiment designed to evaluate the feasibility of FMCheck [de Mello et al. 2012]. Table IV shows the main differences between both trials based on the dimensions and elements proposed by Gómez et al. to classify replications in SE experiments [Gómez et al. 2014]. In this sense, the trial presented in this paper can be considered a *changed-protocol/experimenters* replication of the first trial, as shown in Table 4. Following subsections describe the second trial plan, highlighting the differences from the first trial plan, its execution and results obtained.

Table 4. Comparison between the Characterization of the Dimensions/ Elements from both trials.

Dimension	Element	1st Trial vs. 2nd Trial

Operationalization	Cause	=
	Effect	=
Population	Subjects properties	=
	Objects properties	=
Protocol	Design	≠
	Experimental objects	=
	Guides	=
	Instruments	=
	Data Analysis Techniques	=
Experimenters	Designer, Trainer, Monitor, Measurer, Analyst	≠

4.1. Goal

Based on the GQM template [Caldeira et al. 1994], the purpose of this study was defined as follows:

- *To analyze:* the conducting of feature model inspections by using *ad-hoc* techniques and FMCheck
- *In order to:* characterize
- *With respect to* their capability of providing *efficiency and effectiveness* to the inspection activities
- *From the perspective of:* Software Engineering researchers.
- *In the context of:* evaluating inspection activities performed by other Software Engineering researchers over feature models from different domains

4.2. Question and Metrics

- *Question:* How much time was dedicated to the inspections?
- *Metrics:* Time dedicated to the inspection, and efficiency of each inspection calculated through the formula (1), where *identified defect* represents the amount defects identified and *total time* represents the inspection time (in minutes) for each inspection observation.
- *Question:* Which inspection technique (FMCheck or ad-hoc) allows the inspectors to detect more defects?
- *Metrics:* Number of defects detected, the effectiveness of the inspection calculated through the formula (2), where *identified defect* represents the amount defects identified, and *total defects* represent the total amount of known defects for each feature model inspected.

$$\textit{Efficiency} = (\textit{identified defects} / \textit{total time}) \times 100 \qquad (1)$$

$$\textit{Effectiveness} = (\textit{identified defects} / \textit{total defects}) \times 100 \quad (2)$$

4.3. Hypotheses

- *$H_0$1:* There is no difference between the efficiency of feature model inspections conducted with FMCheck and with *ad-hoc* inspections.
- *H_A1:* The efficiency of feature model inspections conducted with FMCheck is greater than the efficiency of *ad-hoc* ones.
- *$H_0$2:* There is no difference between the effectiveness of feature model inspections carried out with FMCheck and *ad-hoc* inspections.

- *H_A2:* The effectiveness of feature model inspections carried out with FMCheck is greater than that of *ad-hoc* ones.

4.4. Variables

- *Independent variables:* application domains textually described and represented through feature models using the Odyssey-FEX notation [Blois et al. 2006], subject experience in software engineering projects, subject experience in inspections, previous subject knowledge of the domains used in the study.
- *Dependent variables:* Amount of defects, the amount of false positives, time spent in performing the inspection, efficiency, and effectiveness.

4.5. Analysis Mechanism

The replication of the quasi-experiment adopted the following mechanisms for analyzing the collected data:

- Comparison between results of *ad-hoc* and FMCheck inspections to test the hypotheses.
- Calculation of the time spent on the inspections to check efficiency.
- Calculation of variance of the defects and standard deviation in a view to comparing effectiveness/ effectiveness between *ad-hoc* and FMCheck inspections.
- Elimination of outliers and verification of data normality (Shapiro-Wilk) and homoscedasticity (Levene).
- Application of a nonparametric test (Wilcoxon) or a parametric test (Student's t), according to each case.

4.6. Participants

Table 5 summarizes the main characteristics of the subjects from both trials. In the first trial, the sample was composed of 14 students from COPPE/UFRJ. However, in the presented replication, ten graduate students (representing as much as possible software developers) from a Software Reuse course at the Computer Science Department in UFBA were recruited. One can see subjects from the second trial tend to be more experienced in the industry. It is also the only sample having subjects with some previous experience in performing software inspections. However, as in the first trial, all subjects reported only academic experience with feature modeling, a topic introduced in the software reuse classes.

Table 5. Distribution of Participants by Characteristics.

Description	**1st Trial**	**2nd Trial**
Academic degree		
Undergraduate Students	4	0
Graduate Students	10	10
Participant experience with Software Projects		
Two or more projects in industry	4	8
Only a single project in industry	6	1
Only academic software projects	4	1
Participant experience with specific SE Activities		
Software Inspections	0	5
Feature Modeling	0	0

4.7. Experimental Design

Participants should be asked to inspect the same feature models regarding four application domains (i.e., *mobile devices, hospitality, context-aware mobile applications,* and *library*) through two distinct trials. Before the first trial, to prepare the participants for the execution of *ad-hoc* inspections, the participants should be trained in software inspection and domain description through feature models. Then, each participant should perform *ad-hoc* inspections in artifacts from two domains. Next, the participants should be trained in the use of FMCheck before the second trial. After that, each participant should inspect the two other artifacts (domains which had not been inspected by them in the first trial) applying FMCheck.

Before the first trial, the researchers carried out a comparative analysis of the four models to establish the complexity of each feature model applying the following criteria: *number of features*, the *maximum depth of features*, and the *amount of variability*. Based on these three criteria, we categorize the feature models in two different complexity levels: *normal* complexity and more complex models. Thus, two domains were considered with normal complexity level (*mobile devices* and *library*, S01 and S02, respectively) and the other two models considered more complex (*context-aware mobile applications* and *hospitality*, C01 and C02, respectively) [de Mello et al., 2014]. Then, researchers evenly distributed all four domains to be inspected in both trials by each subject [de Mello et al., 2014]. However, after analyzing the results from the first trial, we observed that the complexity of the domains did not influence the performance of effectiveness/ efficiency in both trials. On the other hand, we noted that only the inspections performed over complex domains had their effectiveness significantly benefited through using FMCheck. Thus, in this trial, we arbitrary set C01 and C02 to be inspected only in the first trial (ad-hoc), while S01 and S02 were configured to be inspected only in the second trial (FMCheck). The package containing the artifacts used in this experiment can be requested to the authors' e-mail.

4.8. Execution

The study was executed in February 2014 in the Computer Science Department in UFBA, starting with the completion of the consent form and the characterization form by 10 participants. Then, participants were trained in feature modeling and the Odyssey-FEX notation. Furthermore, an introductory training (one hour) in software inspection was done, including the guidelines for the execution of the first trial. Different from the first trial, the experimental tasks were randomly assigned to each subject. Each participant received an email containing an inspection package, and all 10 participants answered until the given deadline, reporting the defects detected in each artifact inspected. In the second trial, the participants were trained (one hour) in FMCheck, by explaining each verification item and discussing examples of defects that could be detected with these items. After the training session, each participant received the second trial package (composed by instructions, FMCheck instruments and the feature models to inspect). Again, all participants reported on time the defects detected in each artifact inspected.

4.9. Results

Two researchers reviewed all 40 discrepancy-reports, each one from a distinct research group (UFBA, COPPE/UFRJ). In this context, it is important to emphasize that the reviewer from the group that developed FMCheck (COPPE/UFRJ) performed a blind review from all reported defects. It means that the researcher did not know from which subject/trial came the defects. In the end, 283 discrepancies reported by the subjects were classified as defects, and 63 other were classified as false positives. Table 6 and Table 7 summarize the results collected in the first and second trials, respectively. To calculate the efficiency, the total of defects corresponds to the amount of distinct defects detected in a feature model in both trials.

Table 6. Results of the first trial: ad-hoc inspections.

Participant	Domain	Time (min)	#Defects	Efficiency	Effectiveness
P1	C01	100	4	4.00	9.09
	C02	180	3	1.67	10.34
P2	C01	140	6	4.29	22.73
	C02	120	5	4.17	17.24
P3	C01	90	9	10.00	20.45
	C02	60	6	10.00	24.14
P4	C01	20	7	35.00	15.91
	C02	35	2	5.71	6.90
P5	C01	40	8	20.00	18.18
	C02	55	9	16.36	37.93
P6	C01	35	9	25.71	20.45
	C02	30	11	36.67	34.48
P7	C01	73	3	4.11	6.82
	C02	65	3	4.62	10.34
P8	C01	80	3	3.75	6.82
	C02	75	5	6.67	17.24
P9	C01	43	9	21.00	20.45
	C02	50	10	20.00	34.48
P10	C01	40	4	10.00	9.09
	C02	23	3	13.04	10.34

Table 7. Results of the second trial: FMCheck inspections.

Participant	Domain	Time (min)	#Defects	Efficiency	Effectiveness
P1	S01	60	1	1.67	7.69
	S02	120	5	4.17	11.36
P2	S01	60	3	5.00	23.08
	S02	120	7	5.83	15.91
P3	S01	60	8	13.33	84.62
	S02	95	13	13.68	43.18
P4	S01	30	3	13.33	53.85
	S02	20	6	30.00	13.64
P5	S01	50	7	14.00	76.92
	S02	75	7	9.33	15.91
P6	S01	46	8	17.39	84.62
	S02	35	12	34.29	29.55
P7	S01	55	3	5.46	46.15
	S02	40	4	10.00	9.09
P8	S01	50	4	8.00	53.85
	S02	30	5	16.67	11.36
P9	S01	40	9	22.50	69.23
	S02	49	15	30.61	34.09
P10	S01	43	4	9.30	30.77
	S02	27	9	33.33	20.45

The experimental design applied in this trial did not allow us to perform comparisons between both trials based on absolute metrics (such as the *number of defects*, the *number of discrepancies* and *time*) since different artifacts (domains) were inspected in each trial. Thus, the analyses presented in the following subsections are focusing on analyzing the inspections' efficiency and effectiveness.

4.9.1. Efficiency analysis

Table 8 presents the descriptive statistics of the efficiency obtained by the participants. One can see that close values of means and standard deviations have been achieved in both trials, while no outlier was identified. Since normal (log-normal, Shapiro-Wilk test) and homoscedastic distributions were observed (Levene test), we applied Student-t test (alpha = 95%) to test $H_0$1. As a result, no significant difference regarding efficiency between the inspection trials was observed (*p-value*= 0.26443). As a consequence, hypothesis $H_0$1 could not be refuted.

Table 8. Descriptive Statistics from Distribution of Efficiency in Both Trials.

Trial	N	Mean	StDev	Median	Min.	Max.
Ad-hoc	20	12.84	10.51	10.00	1.67	36.67
FMCheck	20	14.89	10.15	13.33	1.67	34.29

Aiming at understanding in which extent each artifact influenced in the distribution of efficiency obtained in each trial, we compared the distribution of efficiency calculated to each artifact inspected in a trial (C01 x C02; S01 x S01). Since all distributions were normal (log-normal, Shapiro-Wilk test) and homoscedastic (Levene test), we applied Student-t test (1-tailed, matched groups) to perform both comparisons. As a result, we could observe that efficiency on inspecting both C01 and C02 was not significantly different (p-value = 0.2899) but efficiency on inspecting S01 and S02 was significantly different (p-value = 0.0361). Thus, we can infer that the artifacts inspected influenced the efficiency observed in FMCheck inspections.

4.9.2. Effectiveness analysis

Table 9 synthesizes the descriptive statistics from the distributions of effectiveness seen in both trials. In both distributions, no outlier was identified. One can see the mean of effectiveness obtained in *ad-hoc* inspections were less than the half of FMCheck one, although only *ad-hoc* distribution presented close values to the median and mean. Since normal distributions were observed (Shapiro-Wilk test) but not homoscedastic (Levene test), it was decided to apply the Mann-Whitney non-parametric test (alpha = 95%) to test $H_0$2. As a result, it was observed that effectiveness of FMCheck inspection was significantly higher than ad-hoc inspections (*p*-value = 0.0086), rejecting $H_0$2 and accepting HA2. Figure 2 shows the boxplots from both distributions of effectiveness, connecting their medians. The presented analysis of the replication results strengthens the findings of the first trial, confirming the observed behavior of significant difference in effectiveness between both trials, favorable to FMCheck [de Mello et al. 2014].

Table 9. Descriptive Statistics From Distribution of Effectiveness in Both Trials.

Trial	N	Mean	StDev	Median	Min.	Max.
Ad-hoc	20	17.67	9.54	17.24	6.82	37.93
FMCheck	20	36.77	25.97	30.16	7.69	84.62

Aiming at understanding in which extent each artifact influenced in the effectiveness obtained in each trial, we compared the distribution of effectiveness obtained to the inspections of each artifact in a trial (C01 x C02; S01 x S01). Since all distributions were normal (log-normal Shapiro-Wilk test) but equal variances were not observed between C01 and C02 (Levene test), we decided to apply non-parametric Mann-Whitney test to perform both comparisons. We noted that effectiveness of inspecting *ad hoc* C01 and C02 was not different (p-value = 0.1539), while we could evidence that effectiveness obtained on inspecting S01 was significantly higher than effectiveness obtained inspecting S02 (p-value = 0.0090). Thus, we can infer that the artifacts inspected influenced the effectiveness observed in FMCheck inspections.

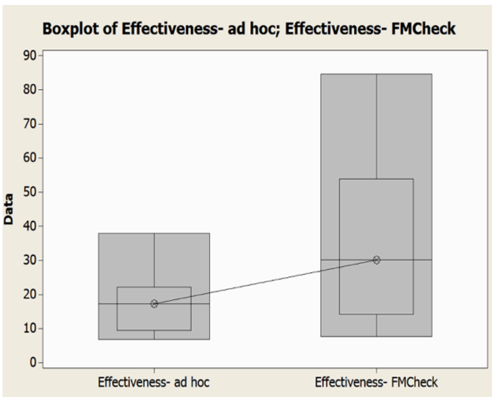

Figure 2. Distributions of Effectiveness in both trials. The line between boxplots represents the median connect line.

4.10. Threats to validity

Similar to the first trial, a small set of artifacts was used in the experimental tasks, which could bias the results to specific domains. However, we expect that reusing the same artifacts in samples with similar characteristics will allow us to better understanding in which extent such domains could influence the results. Again, convenience was applied to establish the experiment sample. However, the differences of background observed between samples can be seen as an opportunity to strength evidence on how FMCheck could help a specific audience having similar characteristics. Finally, we highlight the learning bias on applying FMCheck only in the second trial and the asynchronous execution of the experiment, since subjects performed their tasks out from a controlled environment. However, during the data analysis, we did not observe the incidence of plagiarism.

5. Discussion

The replication presented in this paper strength evidence obtained in the first trial regarding the potential contributions of FMCheck on improving the effectiveness of feature models inspections when compared with *ad-hoc* inspections. At the same time, as typically observed in other inspection techniques, the inspections' efficiency is not improved. Taking into account the different background of the samples (although both are composed of subjects having only theoretical knowledge on feature modeling), aggregating results from both trials may be an opportunity to strength evidence on how FMCheck can be helpful to support reviewers with low experience.

Since inspections were performed over mutually exclusive models in the rounds, comparisons performed between absolute values (*number of defects*, *time*, the *number of discrepancies*) could not be performed. However, we could better understand in which extent each domain could influence the efficiency/ effectiveness obtained in each trial. Although the general results were similar in both trials, we could observe that subjects obtained better results when using FMCheck to inspect the mobile devices (S01) domain than the library (S02) one. Thus, such findings suggest the need for conducting additional replications through applying different arrangements of the same artifacts to better characterizing the actual contributions and limitations of FMCheck.

Different from the first trial, a comprehensive list of known defects of each artifact was available and another research group (distinct from the developers of FMCheck) planned and conducted the experiment, also participating in the results analysis. Another positive aspect observed in the presented replication is regarding all subjects performed all tasks in both trials, allowing the comparison of distributions having the same number of data points. Finally, we highlight that experimental tasks were randomly assigned, characterizing a *full* experiment instead of a *quasi*-experiment (first trial).

6. Conclusions and Future Work

Software product lines have proven its benefits in industrial environments. Thus, to take advantage of these benefits, quality assurance techniques, such as software inspection, should be performed to support the verification of feature models, since such artifact is considered essential for managing knowledge of the specific domains and for reusing assets in different products.

In this work, we presented the replication of a *quasi*-experiment for evaluating efficiency and effectiveness of FMCheck. We observed in the presented replication that FMCheck technique was more effective to inspect feature models than *ad-hoc* inspections, strengthening the results of the first trial. However, the changings on the original experimental design allowed us to observe that FMCheck effectiveness significantly had varied by artifact inspected. On the other hand, it was not observed a significant difference in efficiency between FMCheck and ad-hoc inspections again.

Furthermore, this work is a further step towards evaluating and providing evidence of the feasibility of using FMCheck inspection technique for supporting the inspection of feature models. As future work, the results from both trials will be aggregated and analyzed, and we intend to perform new replications considering different experimental designs and populations. Such further trials will be driven to observe better *in depth* issues that can be addressed to support the improvement of FMCheck efficiency.

Acknowledgement

We would like to thank the students of the 2013.2 Software Reuse course, at UFBA. This work was partially funded by FAPESB (grant BOL0412/2013) and CNPq.

References

Batory, D., Benavides, D. and Ruiz-Cortés, A. (2006) "Automated analysis of feature models: challenges ahead," Communication ACM, vol. 49, pp. 45-47.

Benavides, D., Segura, S., Trinidad, P. and Ruiz-Cortés, A. (2007) “Fama: Tooling a framework for the automated analysis of feature models,” First International Workshop on Variability Modeling of Software-Intensive Systems, pp. 129-134.

Benavides, D., Segura, S. and Ruiz-Cortés, A. (2010) “Automated analysis of feature models 20 years later: A literature review,” Info System, vol. 35, no. 6, pp. 615-636.

Blois, A. P. T. B., de Oliveira, R. F., Maia, N., Werner, C. and Becker, K. (2006) “Variability modeling in a component-based domain engineering process,” 9th International Conference on Reuse of Off-the-Shelf Components, pp. 395-398.

Caldeira, G., Rombach, H. and Basili, V. (1994) “Goal Question Metric Paradigm,” John Wiley Sons, vol. 1.

Cunha, R., Conte, T., Almeida, E. and Maldonado, J. (2012) “A Set of Inspection Techniques on Software Product Line Models,“ 24th International Software Engineering & Knowledge Engineering, pp. 657-662.

de Mello, R., Teixeira, E., Schots, M., Werner, C. and Travassos, G. (2012) “Checklist-based inspection technique for feature models review,” in Sixth Brazilian Symposium on Software Components Architectures and Reuse, pp. 140–149.

de Mello, R. M., Teixeira, E. N., Schots, M., Werner, C. M. L. and Travassos, G. H. (2014) “Verification of software product line artifacts: A checklist to support feature model inspections,” Jornal of Universal Comp. Science, vol. 20, no. 5, pp. 720–745.

de Mello, R. M., Stolee, K. T. and Travassos, G. H. (2015) “Investigating Samples Representativeness for an Online Experiment in Java Code Search” in Nineth ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 1-10.

Gómez, O., Juristo, N. and Vegas, S. (2014) "Understanding replication of experiments in software engineering: A classification," Info and Sw Technology 56.8: 1033-1048.

Kang, K., Cohen, S., Hess, J., Nowak, W. and Peterson, S. (1990) “Feature-Oriented Domain Analysis Feasibility Study,” Technical Report CMU/SEI-90-TR-21.

Northrop, L. M. (2002) “SEI's software product line tenets,” IEEE software, vol 19, no. 4, pp. 32-40.

Rocha, A. R. C., Maldonado, J. C., Weber, K. C. and Travassos, G. H. (2001) “Qualidade de Software - Teoria e Prática,” (in Portuguese) Prentice Hall.

Shull, F., Rus, I. and Basili, V. (2000) “How perspective-based reading can improve requirements inspections,” Computer, vol. 33, no. 7, pp. 73–79.

Souza, I. S., Gomes, G. S. S., Silveira, P. A. M., Machado, I. C., Almeida, E. S. and Meira, S. R. L. (2013) “Evidence of software inspection on feature specification for software product lines,” The Journal of Systems and Software 86: 1172– 1190.

Von Der Massen, T. and Lichter, H. (2004) “Deficiencies in feature models,” Workshop on Software Variability Management for Product Derivation Towards Tool Support.

Zhang, G., Ye, H. and Lin, Y. (2013) “An approach for validating feature models in software product lines,” Journal of Software Engineering, vol. 7, pp. 1-29