Softw Syst Model (2015) 14:1349–1365
DOI 10.1007/s10270-014-0408-2

SPECIAL SECTION PAPER

Evaluating the effort of composing design models: a controlled experiment

Kleinner Farias · Alessandro Garcia · Jon Whittle · Christina von Flach Garcia Chavez · Carlos Lucena

Received: 8 June 2013 / Revised: 29 November 2013 / Accepted: 13 March 2014 / Published online: 6 May 2014

Abstract Model composition plays a key role in many tasks in model-centric software development, e.g., evolving UML diagrams to add new features or reconciling models developed in parallel by different software development teams. However, based on our experience in previous empirical studies, one of the main impairments for the widespread adoption of composition techniques is the lack of empirical knowledge about their effects on developers' effort. This problem applies to both existing categories of model composition techniques, i.e., specification-based (e.g., Epsilon) and heuristic-based techniques (e.g., IBM RSA). This paper, therefore, reports on a controlled experiment that investigates the effort of (1) applying both categories of model composition techniques and (2) detecting and resolving inconsistencies in the output composed models. We evaluate the techniques in 144 evolution scenarios, where 2,304 compositions of elements of UML class diagrams were produced. The main results suggest that (1) the employed heuristic-based techniques require less effort to produce the intended model than the chosen specification-based technique, (2) there is no significant difference in the correctness of the output composed models generated by these techniques, and (3) the use of manual heuristics for model composition outperforms their automated counterparts.

Keywords Model composition effort · Empirical studies · Effort measurement

Communicated by Dr. Jürgen Kazmeier.

K. Farias (✉)
Interdisciplinary Postgraduate Program in Applied Computing (PIPCA), University of Vale do Rio dos Sinos (Unisinos), São Leopoldo, RS, Brazil
e-mail: kleinnerfarias@unisinos.br

A. Garcia · C. Lucena
OPUS Research Group, LES, Informatics Department, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil
e-mail: afgarcia@inf.puc-rio.br

C. Lucena
e-mail: lucena@inf.puc-rio.br

J. Whittle
School of Computing and Communications, Lancaster University, City of Lancaster, UK
e-mail: whittle@comp.lancs.ac.uk

C. von Flach Garcia Chavez
Department of Computer Science, Federal University of Bahia, Salvador, BA, Brazil
e-mail: flach@dcc.ufba.br

1 Introduction

Model composition plays a central role in many software engineering activities, e.g., evolving design models to add new features [1,2]. So, researchers and practitioners have been increasingly concerned with developing effective techniques for composing design models, e.g., [1,3–5]. The term model composition can be defined as a set of tasks that should be performed over two (or more) input models, M_A and M_B, to produce an output intended model, M_{AB}. Given that the elements of the input models may conflict with each other, developers may spend some considerable effort to bring them together. For this reason, both academia and industry have increasingly concerned with developing effective techniques for reducing the composition effort, e.g., Epsilon [4] and IBM Rational Software Architect (RSA) [5].

The existing techniques can be classified into specification-based techniques, such as Epsilon [4], and heuristic-based ones, such as the heuristics supported by the IBM RSA [5].

In the first case, developers explicitly specify both correspondence and composition relations between the elements of the input models, M_A and M_B, to produce M_{AB}. In the second case, developers use a set of predefined heuristics, which "guess" the correspondence and composition relations between the model elements of M_A and M_B before composing them.

However, usually, the use of these heuristic-based composition techniques does not produce the output intended model as would be expected [1,6,7], given that they try to give a solution that is not guaranteed to be optimal, finding a satisfactory solution. Rather, the use of these techniques leads to producing an output composed model, M_{CM}, with inconsistencies. This means that the output composed model (M_{CM}) and the output intended model (M_{AB}) often do not match ($M_{CM} \neq M_{AB}$). These inconsistencies in M_{CM} emerge from conflicting changes between the model elements of the input models (M_A and M_B) that are not correctly resolved. Thus, developers often need to invest some considerable effort to detect and resolve such inconsistencies [2].

On the other hand, the proponents of specification-based techniques claim that explicit composition specifications entail a more systematic way to compose M_A and M_B [8]. Developers expect to reduce the model composition effort using such techniques. Therefore, the conventional wisdom [4] is that a precise specification for composing the elements of the input models is likely to require less effort to produce correctly composed models, i.e., where $M_{CM} = M_{AB}$. However, there is little evidence to confirm if this expectation holds or not. So, developers end up using model composition techniques without any empirical evidence about their effects on the effort to apply them, as well as to detect and resolve inconsistencies.

It is important to highlight that the key motivation for using a particular composition technique is to reduce the developers' effort to produce the output intended model. If a composition technique reduces effort to produce an output composed model and increases the effort to detect and resolve the inconsistencies (or vice versa), then it is quite arguable whether it can be applied in mainstream software projects, given its detrimental impact on the developers' effort.

Although there have been repeated calls for deeper empirical studies focused on investigating the actual effort of using such composition techniques [1,2], little has been done to overcome this ever-present need. Hence, the evaluation of composition effort of composing design models has been largely based on expert reflection and opinion rather than on empirical evidence. We have observed from a series of empirical studies [6] that the adoption of these techniques has been yet based on diverging feedbacks from evangelists rather than on knowledge derived from experimental studies. In [1], France and Rumpe also argue that in the modeling field "the reality is that modelers ultimately rely on feedback from experts to determine "goodness" of their models."

This paper, therefore, reports on a controlled experiment performed with 24 subjects, who use specification-based and heuristic-based composition techniques to evolve design models, more specifically UML class diagrams. In total, the subjects performed 144 compositions of UML class diagrams. We have conducted a comparative analysis about the effort of applying composition techniques, detecting, and resolving inconsistencies in the output composed model. The main results, supported by statistical analysis, suggest that: (1) the chosen specification-based technique requires more developers' effort to produce the intended model than the selected heuristic-based techniques; and (2) there was no significant difference in the correctness of the output composed models generated by the evaluated techniques.

Even though we cannot generalize the results of the experiment to other model composition techniques, this exploratory study represents a first contribution to better understand the potential effects of composition techniques on developers' effort. It is important to highlight that this paper extends [9] by adding new content to the whole parts of the paper, including new discussions about the findings.

The remainder of the paper is organized as follows. Section 2 outlines the main concepts that are going to be used and discussed throughout the paper. Section 3 describes the study methodology. Section 4 discusses the study results. Section 5 describes how threats to validity were minimized. Section 6 compares this work with others, presenting the main differences and commonalities. Finally, Sect. 7 presents some concluding remarks and future work.

2 Background

2.1 Model composition effort

Developers invest effort to fulfill a set of tasks over two (or more) input models, M_A and M_B, to produce an output intended model, M_{AB}. Typically, M_A is the current design model, whereas M_B represents the changes (the delta model) that should be accommodated into M_A to transform it into M_{AB}. That is, M_A is the model to be changed, while M_B has what it is lacking in M_A to transform it into M_{AB}. In practice, developers make use of model composition techniques to insert the upcoming changes, M_B, into the existing base model, M_A. These techniques define the semantic of the model composition and help developers to manipulate M_A and M_B—including their conflicting parts—to produce M_{AB}. Unfortunately, M_{AB} is rarely obtained right away; instead, an output composed model, M_{CM}, with inconsistencies is produced [6]. Thus, we will use the terms *composed model*, M_{CM}, to define the output model with inconsistencies

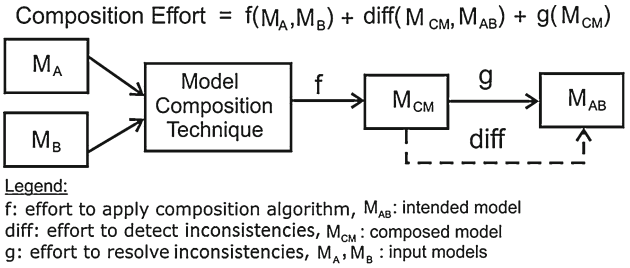


Fig. 1 Overview of model composition effort: an equation

produced by a composition technique and *intended model*, M_{AB}, to express the output model that developers desire to obtain form the composition of M_A and M_B.

As previously mentioned, usually M_{CM} and M_{AB} do not match ($M_{CM} \neq M_{AB}$) because the input models, M_A and M_B, conflict each with other in some way [6,10]. The higher the number of conflicting parts between the M_A and M_B, the higher the probability of inconsistencies in M_{CM} and the more distant it is from the intended model. Figure 1 presents an equation for model composition effort. The equation gives an overview of how composition effort can be measured. The equation makes it explicit that the model composition effort includes: (1) the effort to apply a model composition technique: $f(M_A, M_B)$; (2) the effort to detect model inconsistencies in the output composed model: $diff(M_{CM}, M_{AB})$; and (3) the effort to resolve inconsistencies: $g(M_{CM})$. Once M_{CM} has been produced, the next step is to measure the effort to transform M_{CM} into M_{AB}. If M_{CM} is equal to M_{AB}, then $diff(M_{CM}, M_{AB})$ and $g(M_{CM})$ are equal to zero. Otherwise, $diff(M_{CM}, M_{AB})$ and $g(M_{CM})$ are higher than zero. This study focuses specifically on evaluating the effort to perform these three composition tasks using different composition techniques (described in Sect. 2.3).

2.2 Composition conflicts and inconsistencies

The model elements' properties of M_A and M_B may conflict with each other. Figure 2 shows a simple example of composition conflict. In the base model, the UML class *Researcher* is defined as a concrete class (i.e., *Researcher.isAbstract* = false), whereas the class *Researcher* in the delta model is set as an abstract class (i.e., *Researcher.isAbstract* = true). Then, the developers need to properly answer the question: should class *Researcher* be abstract or not? In this particular case, the correct answer is that the *Researcher* must be abstract (see the intended model also in Fig. 2).

Unfortunately, conflicts that are solved improperly are converted into inconsistencies in M_{CM}, since unexpected values are set to model element's properties. For example, the class *Researcher* is defined as a concrete class, instead of abstract one. Figure 2 shows the class *Researcher* produced by the override and merge algorithms (Sect. 2.3) as a concrete class (*isAbstract* = false) instead of an abstract one (*isAbstract* = true) as would be expected. Note that such inconsistency leads the composed model to be not compliant with the intended model. Two categories of inconsistencies can emerge as follows:

- *Syntactic inconsistency* emerges when a composed model element does not conform to the rules defined in the modeling language's metamodel. For example, a class must have attributes with different names.
- *Semantic inconsistency* arises when the meaning of the elements of the composed model does not match with the meaning of the elements of the intended model. For instance, a class in M_{CM} has an unexpected method or it requires functionality from another class that no longer exists after the composition.

In our study, we focus on semantic inconsistencies because usually they are not automatically identified using model composition techniques. They often require some involvement of software developers. In addition, they are mainly responsible for non-trivial composition problems during the model evolution [2]. Therefore, they may often require more effort and are more detrimental to the correctness of the output model than syntactic inconsistencies [3]. The categories of semantic inconsistencies considered are the following: (1) a model element in M_{CM} is not compliant with the corresponding one in M_{AB}; (2) model elements do not exist in M_{CM}, or exist improperly; (3) model elements are unexpectedly duplicated considering M_{AB}; and (4) there are dangling relationships between classes, i.e., a model element makes reference to other model elements that do not exist. These categories were chosen because they are the most common types of problems faced by developers dealing with model inconsistencies according to previous studies [1,10]. In our study, we also explicitly differentiate each contradicting change (i.e., the conflict) from the result of its incorrect resolution in the output model, i.e., the inconsistency.

2.3 Model composition techniques

The composition techniques used in this study were (1) Epsilon [4], the representative of specification-based techniques, and (2) two heuristic-based composition techniques, namely the IBM Rational Software Architect (RSA) [5], and traditional composition algorithms (TRA) [3,11]. We select these techniques because they support different degrees of automation. While the IBM RSA and the traditional composition algorithms can be applied (semi)automatically and manually, respectively, the Epsilon technique can only be applied semi-automatically. We have chosen the two

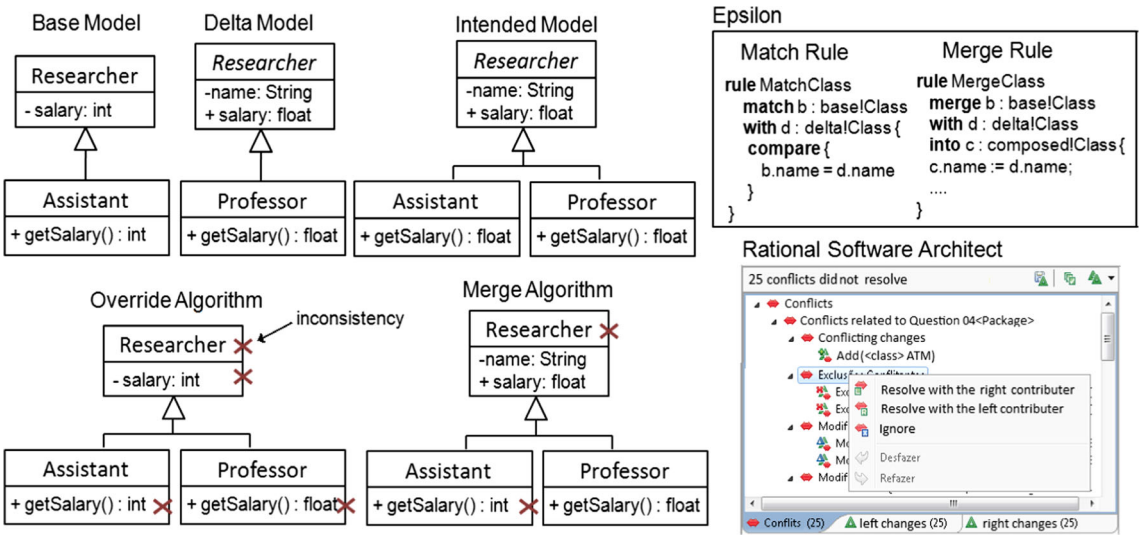


Fig. 2 Illustrative examples

(semi)automated techniques (i.e., Epsilon and IBM RSA) because robust eclipse-based modeling tools support them. This is an essential prerequisite for a controlled experiment like ours. Moreover, IBM RSA is an industry-leading tool, and it is the most widely used tool in the industry [5]. Epsilon is the most representative technique for specification-based composition, as it is a stable and user-friendly tool. The traditional algorithms, such as merge and override, are well explored in the academic literature and have been used to support and guide manual model composition [3,8]. These techniques are better described as follows.

Epsilon (EPS) It provides a hybrid, rule-based language for merging design models [4]. Developers can invest effort to edit a set of match and merge rules before producing M_{AB}. Figure 2 shows an example of these rules. For example, the merge rule specifies that all classes to be composed will have their names equal to the classes of the delta model (i.e., *c.name* := *d.name*). Based on these specifications, developers define how composition relations should be identified.

IBM RSA (RSA) It is one of the most robust modeling tools used in industry [12], and it is characterized as a semiautomated model composition technique. As the Epsilon technique, its use does not ensure that M_{AB} will be always produced. Using the IBM RSA, developers should interactively resolve conflicts before producing M_{AB}. Figure 2 depicts an example of a conflict report. For example, when conflicting changes emerge, developers should decide which changes—from the base model (*Researcher.isAbstract* = false) or from the delta model (*Researcher.isAbstract* = true)—will be inserted into the output composed model.

Traditional algorithms (TRA) These algorithms fall in the category of manual, heuristic-based composition techniques. In particular, we focus on three well-established composition algorithms: override, merge, and union [3,12]. These algorithms were chosen due to three main reasons. First, the model evolution scenarios can be decomposed into one (or more) operation(s) supported by a combination of these algorithms. Second, developers often use these algorithms as guidelines for manually composing OO models [8]. Third, we wish to investigate to what extent the aforementioned automated techniques outperform the use of a classical manual technique in model composition. In the following, we briefly define override and merge algorithms, where we assume two hypothetical input models, M_A and M_B. We say that two elements from M_A and M_B are corresponding if they have been identified as equivalent in the matching process. Matching can be achieved using any number of standard heuristics, such as match-by-name [1].

1. *Override* (direction: M_A to M_B) For all pairs of corresponding elements in M_A and M_B, M_A's elements should override M_B's similar elements. Elements not involved in the correspondence remain unchanged and are inserted into the output model.
2. *Merge* For all corresponding elements in M_A and M_B, the elements should be combined. The combination depends on the element type. In this paper, we only consider classes and interfaces—in this case, the combination adds the operations of M_A's elements to those of M_B. Elements in M_A and M_B that are not equivalent remain unchanged and are directly copied to the output model. In Fig. 2, the override and merge algorithms are applied and two composed models are produced with inconsistencies.

3 Study methodology

This section presents the main decisions underlying the experimental design of the controlled experiment. To begin with, Sect. 3.1 presents the objective and research questions. Section 3.2 systematically states the study hypotheses based on the research questions. Section 3.3 describes the variables and quantification methods considered. Section 3.4 explains the context and subject selection. Section 3.5 introduces the experimental design. Finally, Sect. 3.6 describes the analysis procedures. All these methodological steps are based on practical guidelines about empirical studies described in [13–17].

3.1 Objective and research questions

This study essentially attempts to evaluate the effects of model composition techniques on two variables: the developers' effort and the correctness of the design models. These effects are investigated from concrete evolution scenarios involving compositions of design models so that empirical results can be generated. With this in mind, the objective of this study is stated based on the GQM template [18] as follows:

Analyze *model composition techniques*
for the purpose of *investigating their effects*
with respect to *effort and correctness*
from the perspective of *developers*
in the context of *evolving design models.*

In particular, this study aims at evaluating the impact of model composition techniques on developers' effort and model correctness while evolving design models. Thus, we focus on two research questions:

- **RQ1:** What is the relative effort of composing two input models using specification-based composition techniques with respect to heuristic-based composition techniques?
- **RQ2:** Is the number of correctly composed models higher when using specification-based composition techniques than heuristic-based composition techniques?

3.2 Hypotheses formulation

Hypothesis 1 We conjecture that although specification-based composition techniques provide a more systematic way to compose the input models, their use is not friendly enough to reduce the overall composition effort in practice. Developers may invest much more effort to specify the compositions, and this effort is not converted into a higher number of correctly composed models (than those produced with heuristic techniques). Consequently, we hypothesize that the specification-based composition technique tends to require a higher effort to compose design models rather than the heuristic-based ones. However, it is by no means obvious that this hypothesis holds. Maybe, the specification-based techniques may help developers to match and then compose the input models more quickly; or even they may offer some advantages for expressing complex evolution scenarios. Hence, they may allow developers (1) to accommodate the changes from M_B into M_A properly and (2) to tame the conflicting parts between them more systematically, thereby reducing the overall effort to produce M_{AB}. Therefore, the first hypothesis evaluates whether the specification-based composition technique requires a higher developers' effort than the heuristic-based one. Based on this statement, we state the null and alternative hypotheses as follows:

Null Hypothesis 1, $\mathbf{H_{1-0}}$: The specification-based composition technique requires less (or equal) effort than the heuristic-based composition technique to produce M_{AB} from M_A and M_B.
$\mathbf{H_{1-0}}$: $\text{Effort}(M_A, M_B)_{\text{Specification}} \leq \text{Effort}(M_A, M_B)_{\text{Heuristic}}$
Alternative Hypothesis 1, $\mathbf{H_{1-1}}$: The specification-based composition technique requires more effort than the heuristic-based composition technique to produce M_{AB} from M_A and M_B.
$\mathbf{H_{1-1}}$: $\text{Effort}(M_A, M_B)_{\text{Specification}} > \text{Effort}(M_A, M_B)_{\text{Heuristic}}$

By testing this first hypothesis, we evaluate if the type of composition technique plays a critical role to minimize the effort that developers should invest to produce the output intended model, thereby generating empirical evidence about how these techniques accommodate upcoming changes from M_B to M_A. Knowing which technique is less effort consuming, developers can rationally (not intuitively) choose the most appropriated one for a particular evolution scenario. This strategy is a more effective one than the usage of the techniques based exclusively on feedbacks of experts. We refine this hypothesis in another *three sub-hypotheses* ($H1_2$, $H1_3$, and $H1_4$). A complete formulation of these hypotheses can be seen in Table 1.

Hypothesis 2 As previously mentioned, developers may produce an output composed model with inconsistencies from two input models. Depending on the amount of inconsistencies in M_{CM}, developers tend to invest an increased effort to derive M_{AB} from M_{CM}, i.e., effort to detect and resolve inconsistencies. If the effort is too high, then the use of the technique in issue can be doubted. The main reason for using a specification-based composition technique is the fact that it can produce a higher number of correctly composed models than its counterpart by allowing developers to freely express how the input models will be compared and inte-

Table 1 Tested hypotheses

Null hypothesis	Alternative hypothesis
$H1_{1-0}$: $Effort(M_A, M_B)_S \leq Effort(M_A, M_B)_H$	$H1_{1-1}$: $Effort(M_A, M_B)_S > f(M_A, M_B)_H$
$H1_{2-0}$: $f(M_A, M_B)_S \leq f(M_A, M_B)_H$	$H1_{2-1}$: $f(M_A, M_B)_S > f(M_A, M_B)_H$
$H1_{3-0}$: $diff(M_{CM}, M_{AB})_S \leq diff(M_{CM}, M_{AB})_H$	$H1_{3-1}$: $diff(M_{CM}, M_{AB})_S > diff(M_{CM}, M_{AB})_H$
$H1_{4-0}$: $g(M_{CM})_S \leq g(M_{CM})_H$	$H1_{4-1}$: $g(M_{CM})_S > g(M_{CM})_H$
$H2_{1-0}$: $Cor(M_{CM})_S \leq Cor(M_{CM})_H$	$H2_{1-1}$: $Cor(M_{CM})_S > Cor(M_{CM})_H$
$H2_{2-0}$: $Rate(M_{CM})_S \geq Rate(M_{CM})_H$	$H2_{2-1}$: $Rate(M_{CM})_S < Rate(M_{CM})_H$

Effort Effort to compose the input models (RQ1), *S* specification-based composition technique
f Effort to apply the composition technique (RQ1), *H* heuristic-based composition technique
diff Effort to detect inconsistencies (RQ1), *g* effort to resolve inconsistencies (RQ1)
Cor Correctness of the composition (RQ2), *Rate* inconsistency rate of the composed model (RQ2)

grated. Using match and merge rules, for example, developers can express how each class and relationship present in two input class diagram should be compared and then merged. On the other side, the heuristic-based techniques guess the equivalence between the input models and then integrate them. Although the heuristics can compose the input models improperly, it is not clear whether in fact offering them a do-it-yourself model composition approach will reduce the developers' effort. If the developers do not precisely define the comparison and integration rules, the structure of the composed models may be affected in different ways, e.g., creating unexpected interdependencies between the model elements. Nevertheless, the developers may have some difficulties for expressing how the composition should be done, we conjecture that they are able to create adequate comparison and merge rules, given the need to express the evolution scenarios with rules. Thus, the heuristic-based one cannot outnumber the number of correctly composed models. We hypothesize that the specification-based composition technique will produce a higher number of correctly composed models than the heuristic-based ones. Therefore, the second hypothesis evaluates whether the specification-based composition technique can in fact help developers to improve the correctness (Cor) of the output model when compared to the use of heuristic approaches. With this in mind, we formulate the null and alternative hypotheses as follows:

> **Null Hypothesis 2**, $\mathbf{H_{2-0}}$: The specification-based composition technique produces a lower (or equal) number of correctly composed models than the heuristic-based composition technique.
> $\mathbf{H_{2-0}}$: $Cor(M_{CM})_{Specification} \leq Cor(M_{CM})_{Heuristic}$
> **Alternative Hypothesis 2** , $\mathbf{H_{2-1}}$: The specification-based composition technique produces a higher number of correctly composed models than the heuristic-based composition technique.
> $\mathbf{H_{2-1}}$: $Cor(M_{CM})_{Specification} > Cor(M_{CM})_{Heuristic}$

The composition correctness is influenced by the presence (or not) of the inconsistencies in the output composed model. Thus, we investigate whether the specification-based technique entails (or not) a lower inconsistency rate than the use of the heuristic-based techniques. Table 1 states this new elaborated hypothesis. By testing the second hypothesis (H2), we produce empirical knowledge about the impact of the composition techniques on both the model correctness and the inconsistency rate. Knowing which technique is more error-prone, developers can choose the technique more properly.

3.3 Study variables

Dependent variables in the first hypothesis The dependent variables of the first hypothesis are the effort variables present in Table 1, including the overall effort to compose two input models, $Effort(M_A, M_B)$, the effort to apply the model composition techniques, $f(M_A, M_B)$, the effort to detect inconsistencies $diff(M_{CM}, M_{AB})$, and the effort to resolve the inconsistencies, $g(M_{CM})$. We measure these variables in minutes. The main reason why we have investigated these variables is that they are the most important tasks performed by developers to integrate two input models in realistic settings [2,6]. The computation of these variables allows us to study the impact of the independent variable on each one of them. By comparing the values (in minutes) assumed by these variables, we can also grasp how a composition technique outnumbers the other one considering a particular task.

Dependent variables in the second hypothesis The dependent variables of the second hypothesis are the model correctness and the inconsistency rate. Considering the first, the output composed model produced is *correct* ($H2_1$) if it is compliant with the change requests, i.e., $M_{CM} = M_{AB}$ where the full correctness of a composition is assured. We compare M_{CM} with M_{AB} (our "reference intended models") considering the actual model elaborated by the actual developers of those systems, from which the input models were extracted. The composed model may be rated as either *correct* or *incorrect*. Note that a composed model with one of the incon-

Table 2 Tasks of the evolution scenarios

Task	Models	Required changes to the base model
1	Oil extraction	*Add* one class, one method, and one relationship *Modify* one class from concrete to abstract
2	Car system	*Remove* two methods and *modify* the direction of a relationship
3	ATM	*Add* two classes and *refine* two classes from one *Remove* this last class
4	Supply chain	*Add* two classes and one relationship
5	Financial	*Remove* one class and *add* two methods to a particular class *Refine* two classes from one and remove the last one. *Remove* one relationship
6	Simulation of extraction	*Modify* the direction of five relationships *Modify* the name of two methods

sistencies previously described (Sect. 2.2) was deemed as *incorrect*. We also investigate the inconsistency rate of the incorrectly composed model ($H2_2$). It represents the ratio of the number of inconsistencies of a composed model by its number of model elements. That is, it quantifies the amount of composition inconsistencies ($H2_2$) divided by the total number of elements in the composed model. It allows to calculate the density of composition inconsistencies in the output composed model. By comparing the inconsistency rate produced, we can understand which techniques are more effective for producing models closer to the output intended model. That is, this metric makes it possible to assess the difference between the inconsistency rates in M_{CM} produced with specification-based and heuristic-based techniques.

Independent variable The independent variable of hypotheses 1 and 2 is the use of the model composition technique. As previously mentioned, we control the use of specification and heuristic-based techniques to better understand their impact on the dependent variables previously mentioned.

3.4 Context and subject selection

The subjects used three composition techniques (i.e., Epsilon, IBM RSA and traditional algorithms) to perform six evolution scenarios and were familiar neither with such scenarios nor with the design models used. Table 2 shows the evolution scenarios describing typical tasks in which developers should evolve design models. It is important to highlight that the evolution represents the cases where the subjects are not the initial designers of the models. The design models used were fragments of industrial models captured from different application domains, including financial and simulation of oil extraction.

The experiment was conducted with 16 professionals from Brazilian companies and 8 students with professional experience. The professionals held a Master's degree and Bachelor's degree (or equivalent) and had a considerable knowledge of software modeling and programming. The students were also invited to participate in the experiment so that we could have subjects with different backgrounds and levels of expertise. They were from two postgraduate programs in Computer Science at two Brazilian universities, namely Pontifical Catholic University of Rio de Janeiro (PUC-Rio) and the Federal University of Bahia (UFBA). These students attended two courses with the following themes: "empirical studies in software engineering" (PUC-Rio) and "software evolution" (UFBA). The experiments were part of the two postgraduate courses (at PUC-Rio and UFBA) and were performed as practical laboratory exercises. Each participant was exposed to the same level of training about the model composition techniques being assessed.

3.5 Experimental design

The experimental design of this study is characterized as a randomized complete block one [13] with three treatments, i.e., the use of the three techniques. The study had a set of activities that were organized in three phases (see Fig. 3). The subjects were randomly assigned and equally distributed to the treatments, following a within-subjects design in which all subjects serve in the three treatments [13]. In each treatment, the subjects used a model composition technique to carry out two experimental tasks (Table 2), totaling six tasks performed. Therefore, the experiment design was, by definition, a balanced design. Figure 3 shows through an experimental process how the three phases were organized. The subjects individually performed all activities to avoid any threat to the experimental process. The activities are further described as follows.

Training All subjects received training to ensure they acquired the needed familiarity with each model composition technique.

Apply the techniques The participants were encouraged to compose M_A and M_B based on a description of changes

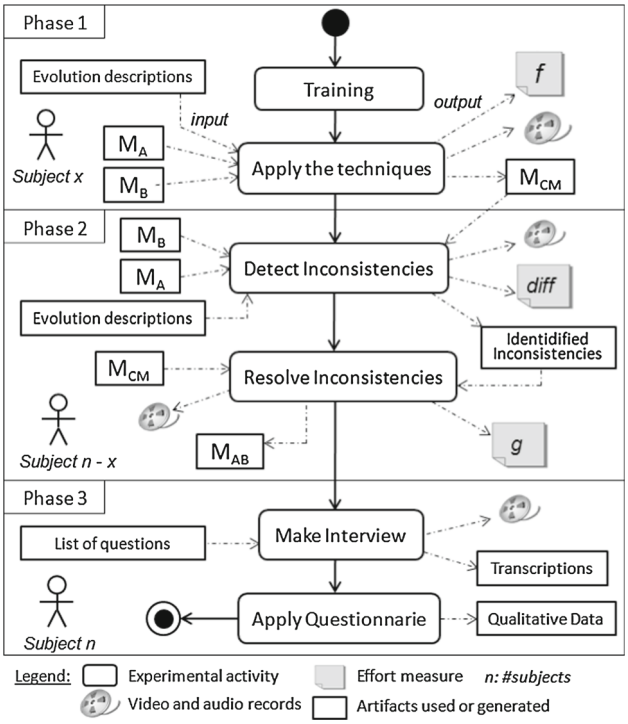


Fig. 3 The experimental process

(Table 2) that defines how the model elements of M_A were changed. Note that M_B (delta model) was ready. The measure of application effort (time in minutes) was collected during this activity. In addition, the composed model, video and audio records represent the outputs of this activity. Each subject performed this task six times. The video and audio records were later used during the qualitative analyses. It is important to point out that a participant (subject x) produced M_{CM} in the first phase; in the second phase, other participants (subject n-x) detected and resolved the inconsistencies in M_{CM} to produce M_{AB}.

Detect inconsistencies Subjects reviewed M_{CM} to detect inconsistencies. To this end, they checked whether M_{CM} had the changes described in the evolution descriptions and if the contradicting changes between M_A and M_B were correctly addressed. As a result of this activity, we have the measure of detection effort (time in minutes), video and audio records, and a list of inconsistencies identified.

Resolve inconsistencies The subjects resolved the inconsistencies previously localized to produce M_{AB}. The resolution effort was also measured (time in minutes), and the video and audios were recorded.

Make interview and Answer questionnaire Subjects reflected on their experience on model composition during the experiment through semi-structured interviews. These interviews helped us to enrich the body of qualitative data collected. The subjects also filled out a questionnaire. This allowed us to collect their background (i.e., their academic background and work experience) and apply some inquisitive questions.

Material The models used in our study were UML class diagrams with by about 8 classes and 7 relationships. This medium size of the models was essential to perform a controlled study like this and to be in compliance with recommendations from previous work. For example, Asklund et al. [13] recommends that software changes should be as small as possible so that the number of conflicts remains small. In addition, given the time constraints of controlled experiments, the subjects could not be exposed to very large models.

3.6 Analysis procedures

Quantitative analysis We performed descriptive statistics to analyze its normal distribution [13,19] and statistical inference to test the hypotheses. The level of significance of the hypothesis tests was $\alpha = 0.05$. The analyses were carried out to test the hypotheses both individually for each experiment task and across all experiment tasks. To test $H1_1$ (and its *sub-hypotheses*), we applied the nonparametric *Wilcoxon signed-rank test* for the six tasks. This test is similar to the *t* test, but does not require two separate sets of independent and identically distributed samples. Note that we have a same subject design. As a result, our samples are dependent. Moreover, the nonparametric *Friedman* ANOVA *test* [13,19] was also applied to reduce some potential threats to the validity of statistical conclusions. To test $H2_1$, we applied the McNemar's test for marginal homogeneity. To test $H2_2$, we consider the inconsistency rate produced during the evolution scenarios. As in H1, we also applied the Wilcoxon signed-rank test and Friedman test.

Qualitative analysis Qualitative data were collected from some sources: questionnaire, audio/video records, and transcriptions, think aloud comments and interviews. This helped us to obtain potentially some complementary evidence to explain the quantitative results and then derive concrete conclusions from a chain of evidence, which are formed from the systematic alignment of the quantitative and qualitative data.

4 Study results

This section analyzes the data set obtained from the experimental procedures described in Sect. 3. Our findings are derived from both the numerical processing of this data set and the graphical representation of interesting aspects of the gathered results. Section 4.1 elaborates on the gathered data

Table 3 Descriptive statistic for the composition effort

	Effort			f			Diff			g		
	TRA	RSA	EPS	TRA	RSA	EPS	TRA	RSA	EPS	TRA	RSA	EPS
N	46	46	46	46	46	46	46	46	46	46	46	46
Min	5	5	9	2	3	4	1	1	1	0	0	0
25th	7	11	14	4	6	8.7	2	2	3	0	0	0.5
Med	11	14	21	6	8	12	3	4	4.5	0.5	2	3
75th	18	24	34	9	11	17	5.2	8	8.7	4	7	9
Max	31	66	114	25	22	39	11	22	38	9	22	38
Mean	13.3	18.2	29.1	7.2	9.0	14.8	3.9	5.3	7.7	2.1	3.8	6.6
SD	6.9	11.0	23.3	4.4	4.2	8.8	2.4	4.4	8.2	2.9	5.1	9.1

N #compositions, *Min* minimum, *Med* median, *Max* maximum, *SD* standard deviation, *TRA* traditional, *RSA* rational software architect, *EPS* Epsilon

in order to test the first hypothesis (H1). Section 4.2 discusses the collected data related to the second hypothesis (H2). Section 4.3 presents some additional observations.

4.1 RQ1: effort and composition techniques

Descriptive statistics This section discusses the collected data with respect to the impact of composition techniques on the developers' effort. For this, we compute descriptive statistics to grasp the data distribution, including its main trends and dispersions. Table 3 shows the descriptive statistics of the collected data. Note that these statistics are calculated based on 138 compositions, i.e., with 46 compositions using each one of the composition techniques (IBM RSA, Epsilon, and traditional algorithms). The main finding is that *the developers tend to invest less effort to produce M_{AB} using heuristic-based techniques rather than the specification-based technique*. The data show that they spent less effort to apply the composition techniques, f(M_A, M_B), detect inconsistencies, diff(M_{CM}, M_{AB}), and resolve inconsistencies, g(M_{CM}). The traditional algorithms required less effort than the IBM RSA, which in turn required less than the Epsilon. This is an interesting finding because the common sense would be otherwise, i.e., developers would invest less effort using an (semi)automated technique, Epsilon and IBM RSA.

Regarding the median of the general effort, it grew significantly from 11 to 14 and 21 using RSA and Epsilon, respectively. This superior effort represents an increase by about 27.27 and 90.9 %. This upward trend was also observed in *f*, *diff*, and *g*. Considering the mean of effort computed, this evidence was still stronger. The general effort increased from just over 13 min in the traditional algorithms to 18.26 min in the IBM RSA, reaching almost 30 min in the Epsilon. This represents a rise of 36.88 and 118.66 %, respectively. This result, therefore, demonstrates that the developers tend to invest less effort with heuristic-based techniques than specification-based one. The next step aims at scrutinizing whether the evidence statistically significant to reject the null hypotheses (H_{1-1}, H_{1-2}, H_{1-3}, and H_{1-4}) stated in Sect. 3.2.

Hypothesis testing We also performed statistical tests to evaluate whether in fact the measures of Effort(M_A,M_B), f(M_A, M_B), diff(M_{CM}, M_{AB}), and g(M_{CM}) are *statistically significant*. We have hypothesized that the specification-based technique tends to require a higher effort than its counterpart (Table 1). So, the test of the mean difference between the measures of the composition techniques (i.e., IBM RSA, Epsilon, and traditional algorithms) will be performed as one-tailed test. To indicate a true significance, we considered the significance level at 0.05 level (p value ≤ 0.05), as previously mentioned.

Since the Shapiro–Wilk test [13] indicated deviations from normality, the Wilcoxon signed-rank test and Friedman test were applied. While the Wilcoxon test allowed us to realize a pairwise comparison of the distributions, Friedman test allowed us to check whether there exist significant differences among the three techniques under investigation.

Wilcoxon test We test H1 (and its sub-hypotheses stated in Sect. 3.2) to evaluate the RQ1 in the six experimental tasks (Table 2). Table 4 shows the p values for the pairwise comparison. Bold p values highlight statistically significant results, i.e., p value < 0.05. They indicate the rejection of the respective null hypothesis. The main feature is that the general composition effort (f, diff, and g) using heuristic-based techniques was significantly lower than using (semi)automated techniques in all cases. Still, using the traditional algorithms, this significance is higher. Thus, we can reject the H1's null hypotheses (and its $H1_{1-0}$, $H1_{2-0}$, $H1_{3-0}$, and $H1_{4-0}$). For example, in row 1 in Table 4, for measure *General Effort*, between RSA and EPS, the W is negative (-544) and p value is less than 0.05 ($p = 0.001$). This means that the composition effort using the IBM RSA is significantly lower than one

Table 4 The results for Wilcoxon test

Task	S	General effort			$f(M_A, M_B)$			$diff(M_{CM}, M_{AB})$			$g(M_{CM})$		
		A	B	C	A	B	C	A	B	C	A	B	C
All	*p*	**0.005**	**0.0001**	**0.001**	**0.02**	**0.0001**	**0.0003**	**0.03**	**0.0003**	0.08	**0.01**	**0.0003**	**0.04**
	W	−420	−900	−544	**−277**	−834	−588	−233	−533	−186	−261	−423	−248
1	*p*	0.33	0.5	0.5	0.42	0.40	0.3628	0.14	0.5	0.39	0.46	0.39	0.30
	W	6	0	0	−4	5	6	16	−1	4	−2	−4	−7
2	*p*	**0.01**	**0.003**	0.14	0.23	**0.007**	**0.0342**	**0.01**	0.22	0.23	0.08	0.05	0.22
	W	−32	−36	−16	−12	−34	−27	−21	−8	8	−14	−24	−10
3	*p*	0.28	**0.01**	0.13	0.37	**0.01**	0.1548	0.27	0.05	0.12	0.23	0.06	0.12
	W	−8	−21	−14	−4	−26	−16	−8	−20	8	−8	−10	12
4	*p*	0.5	**0.01**	**0.01**	0.29	**0.01**	**0.0171**	0.29	0.06	**0.03**	0.5	**0.01**	**0.04**
	W	−1	−28	−26	−3	−28	−26	3	−19	−22	0	−21	−17
5	*p*	**0.01**	**0.007**	0.97	0.07	**0.003**	**0.0177**	**0.02**	0.8	0.19	0.27	0.43	0.5
	W	−26	−36	−20	−18	−36	−31	−11	−25	−11	−8	−3	−1
6	*p*	**0.04**	**0.03**	0.42	0.21	0.07	0.1094	0.06	**0.01**	0.11	**0.04**	0.12	0.42
	W	−21	−23	3	−9	−18	−13	−12	−28	15	−17	−28	28

W sum of signed ranks, *RSA* IBM rational software architect, *EPS* Epsilon, *TRA* traditional algorithm, *A* TRA versus RSA, *B* TRA versus EPS, *C* RSA versus EPS, *p p* value, *S* statistics

using Epsilon. Still in row 1, only one null hypothesis was not rejected in just one case: the effort to detect inconsistencies considering the IBM RSA and Epsilon (p value $= 0.0891$). This means that the subjects did not spend substantially different effort to detect inconsistencies in IBM RSA and Epsilon. Therefore, our initial intuition that the specification-based technique would not reduce the composition effort was confirmed.

Friedman's test Given this result, we were encouraged to apply the Friedman's test to eliminate threats to statistical conclusion validity. This test also confirmed the previous conclusions. The results are shown in Table 5. Again bold p value (<0.05) means that there is a significant difference between the mean ranks in repeated measures of effort. Hence, there is sufficient evidence to reject the null hypothesis, and conclude that there is a difference between the composition efforts at the 0.05 level of significance. For example, in row 1, a chi-square value of 26.21 with $p < 0.05$ indicates a statistically significant difference in the effort measures associated with the three techniques.

Table 5 The Friedman test for the composition effort

Task	Statistics	Effort	$f(M_A, M_B)$	$diff(M_{CM}, M_{AB})$	$g(M_{CM})$
All	*p* value	**0.0001**	**0.0001**	**0.0048**	**0.0017**
	χ^2	26.21	26.64	10.66	12.76
1	*p* value	0.7682	0.8135	0.5690	0.3977
	χ^2	0.8571	0.4	1.1515	1.931
2	*p* value	**0.0048**	0.0789	0.0789	0.1495
	χ^2	9.75	5.25	5.12	3.931
3	*p* value	0.1916	0.1916	0.4861	0.3046
	χ^2	3.630	3.630	1.68	2.5454
4	*p* value	**0.0084**	**0.0036**	**0.0272**	**0.0207**
	χ^2	8.615	9.333	6.333	7.5238
5	*p* value	**0.0099**	0.0024	0.0024	1
	χ^2	8.968	10.516	10.51	0
6	*p* value	0.0854	**0.0272**	**0.0207**	**0.0003**
	χ^2	5.429	6.231	7.6923	12.074

χ^2 Friedman's chi-square, $\alpha = 0.05$

4.2 RQ2: correctness and composition techniques

Descriptive statistics This section analyzes the collected data with respect to the impact of composition techniques on the model correctness and the inconsistency rate. For this, we also compute descriptive statistics to grasp the data distribution, including its main trends and dispersions, as previously done. Figure 4 shows the correctness of the compositions generated using the three techniques: traditional algorithms, Epsilon, and IBM RSA during the six experimental tasks. The *y*-axis represents the proportions of M_{AB} achieved by the number of compositions realized in each task using each composition technique, while the *x*-axis consists of the experimental tasks. Thus, the histogram shows how the correctly composed model happened throughout the experimental tasks.

The main outstanding feature is the lack of a distribution pattern of the proportions of correctly composed models in the tasks. For example, in task 1, TRA produced a lower

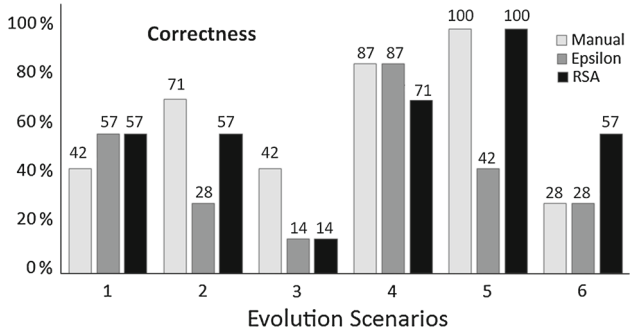


Fig. 4 The correctness of the output composed model

proportion of correctly composed models than RSA and EPS. That is, the intended model was generated in 42 % of the cases in TRA, whereas 57 % of the cases in RSA and EPS. On the other hand, in task 2, TRA outnumbers RSA and EPS. It produced the intended model in 71 % of the cases, while EPS and RSA produced 28 and 57 % of the cases, respectively. Although TRA has obtained low measures in task 3 in comparison with task 2 (a decrease from 71 to 42 %), it still got a superior value compared to EPS and RSA, i.e., the value by about three times higher than the measure of EPS and RSA, comparing 42 and 14 %. Moreover, TRA and EPS had an equal proportion of correctly composed model in task 4, presenting an increase of around 20 % considering RSA.

On the other hand, in task 6, this superiority was reversed. RSA got double the value than TRA and EPS, comparing 28 and 57 %. In task 5, the superiority of TRA and RSA considering EPS was evident. Still subjects obtained the intended model using TRA and RSA in all composition cases, while less than half of the cases in EPS. We have observed that TRA got a higher number of intended models than RSA and EPS. The subjects produced the intended model in 61 % of the compositions using TRA against 59 and 42 % using the RSA and Epsilon technique, respectively. Two interesting insights were that (1) the composition techniques require different effort in front of the categories of evolution changes, and (2) the specification-based technique does not guarantee superiority in terms of correctness in comparison with the heuristic-based techniques.

Table 6 shows the descriptive statistics of the inconsistency rate of the composed models. Our initial expectation was that the specification-based technique would minimize the inconsistence rate, whereas also get lower measures than the heuristic-based techniques. However, this expectation was not confirmed. We have observed that the inconsistency rate was similar in specification-based and heuristic-based techniques in most cases. This means that developers will not produce correctly composed model using a technique based on composition specifications. Rather, the output models will have equal (or even more) inconsistency rate.

Table 6 The descriptive statistics for the inconsistency rate

	N	Med	75th	Max	Mean	SD
TRA	46	0	0.31	1.63	0.26	0.45
RSA	46	0	0.425	1.22	0.21	0.29
EPS	46	0.47	0.78	5.22	0.58	0.88

Table 7 The McNemar test for the inconsistency rate

Task	Comparison	χ^2	p value
All	TRA versus RSA	0.27	0.606
	TRA versus EPS	0.75	0.387
	RSA versus EPS	0	1

For example, on average, EPS produced a higher inconsistency rate than TRA and RSA. In general, the mean of the inconsistency rate in Epsilon is two times higher than one TRA and RSA, increasing by about 123 and 176 %, respectively. This suggests that the inconsistency rate has favored TRA and RSA in comparison with EPS in most cases. This implies that to some extent the number of inconsistencies is decreased whenever the composed model is produced by TRA and RSA.

Hypothesis testing RQ2 evaluates whether the specification-based techniques assure a higher number of correctly composed model than the heuristic-based techniques. We test $H2_1$ (and $H2_2$) to investigate RQ2. We apply the McNemar test to test $H2_1$. Table 7 shows the chi-square statistic and p values for the pairwise comparisons. In all cases, the p value was large ($p > 0.05$), so the null hypothesis of $H2_{1-0}$ cannot be rejected. Although the p value to the six tasks is not shown in the table, the p value took values greater than 0.05 in the six tasks. This implies that there is no significant difference between the proportions of correctly composed model produced by the composition techniques.

We test $H2_2$ by applying the Wilcoxon test. Table 8 depicts the pairwise p values for each measure. Bold p values point out the statistically significant results. They also indicate the rejection of the null hypothesis. Note that the sum of signed ranks (W) shows the direction in which the result is significant. For example, in row 2, W is negative (-250) and p value is lower than 0.05 ($p = 0.0301$) for measure between TRA and EPS. This means that the inconsistence rate for TRA is significantly lower than in EPS. RSA also obtained an inconsistence rate significantly lower ($p = 0.001$) than EPS. For instance, in row 1, the W is negative (-5) and p value is higher than 0.05 for the inconsistency rate between TRA and RSA. This means that the inconsistency rate for TRA is lower, but no significantly lower than RSA.

Table 8 The Wilcoxon test for the inconsistency rate

Tasks	Statistic	Inconsistency rate		
		TRA versus RSA	TRA versus EPS	EPS versus RSA
All	*p* value	0.4851	**0.0301**	**0.0011**
	W	−5	−250	344
1	*p* value	0.2188	0.2188	0.5000
	W	7	7	−1
2	*p* value	0.3750	0.2188	0.0781
	W	2	−9	15
3	*p* value	0.2002	0.1094	0.1355
	W	−9	−16	14
4	*p* value	0.5000	0.5000	0.2071
	W	−1	1	−4
5	*p* value	0.5000	0.1875	0.1250
	W	1	−6	8
6	*p* value	0.1982	0.1094	**0.0469**
	W	9	−16	17

W sum of signed ranks

Table 9 The Friedman test for the inconsistency rate

Task	Statistics	Rate
All	*p* value	**0.0258**
	χ^2	7.314
1	*p* value	0.7682
	χ^2	0.4210
2	*p* value	0.0854
	χ^2	4.666
3	*p* value	0.4861
	χ^2	1.407
4	*p* value	0.7682
	χ^2	0.666
5	*p* value	0.4861
	χ^2	2
6	*p* value	0.2366
	χ^2	3.3076

χ^2 Friedman's chi-square, $\alpha = 0.05$

These results also encouraged us to apply the *Friedman test.* We obtained a chi-square value (χ^2) of 7.314 with a *p* value = 0.0258, which is lower than 0.05 hence is significant. This means that there exists a significant difference between the inconsistency rate using TRA, RSA, and EPS. However, considering each experiment task, the results did not take significance (i.e., $p > 0.05$). This means that a technique did not significantly outperform the other two ones. For example, in task 1 in Table 9, the Chi square value (χ^2) of 0.4210 with a *p* value = 0.7682 indicates that there exists no significant difference between the three techniques in terms of inconsistency rate.

4.3 Additional observations

We also investigated whether the aforementioned results could be explained based on some information collected during the interviews and the analysis of the qualitative data, i.e., video and audio records. First, the subjects mentioned that they often had some additional difficulties to match and compose the input model elements using the specification-based composition techniques. They claimed that it is particularly challenging for them to precisely define the match and merge rules, given the change task's semantics at hand. More specifically, this problem was particularly observed in compositions dominated by relations of the type one-to-many (1:N) or many-to-many (N:N) between the input model elements. The following extract from the interview also illustrates the difficulty associated with understanding the scope of elements involved to specify a composition: "…express the changes in the match and merge rules is boring…because all overlapping parts of the two input models should be analyzed…this is not a trial task."

Second, the IBM RSA tool shows the commonalities and differences between the input models in multiple, partial views. This strategy jeopardizes the subjects to create a "big picture view" of the intended model for each composition. The following extract confirms this observation: "I have to check more than three views to complete something…it is very complicated when more complex changes happened… because I have to mentally infer a complete, unique view. On the other hand, the strict use of the traditional algorithms is much more intuitive and allows me to work freely and closer to the manner that I think about model composition."

Finally, we have observed that: (1) the model composition techniques should be more intuitive and flexible to express different types of changes such as addition, removal, modification, and refinement of model elements; (2) the techniques should represent the conflicts between the input models in more innovative views; (3) the emerging composition techniques should be a mixture of specification-based and heuristic-based techniques; and (4) the heuristic-based techniques consumed less effort and were more effective than their counterparts. This suggests that the tools for specification-based techniques may be very rigid and need more flexibility so that, for example, developers can adjust the composition specification considering their experience.

5 Threats to validity

This study has a number of threats to validity that range from statistical conclusion validity, construct, internal, and external threats. This section discusses the strategies used for managing these threats.

5.1 Statistical conclusion validity

We minimized this threat by checking whether the independent and dependent variables were submitted to suitable statistical methods. The evaluation checked (1) whether the presumed cause and effect covary and (2) how strongly they covary [14]. Considering the first inferences, we may improperly conclude that there is a causal relation between the variables when, in fact, they do not. We may also incorrectly state that the causal relation does not exist when, in fact, it exists. With respect to the second inference, we may incorrectly define the magnitude of covariation and the degree of confidence that the estimate warrants [20].

Covariance of cause and effect We minimized the threats to the causal relation between the research variables studying the normal distribution of the collected sample. Thus, it was possible to verify whether parametric or nonparametric statistical methods could be used (or not). For this purpose, we used the Kolmogorov–Smirnov and Shapiro–Wilk tests [19] to check the normal distribution of the data. Hence, we are confident that the test statistics were applied correctly, as the assumptions of the statistical test were not violated.

Statistical significance We test all hypotheses considering the significance level at 0.05 level ($p \leq 0.05$). In addition, we followed some general guidelines to improve conclusion validity [21]. First, we tried to obtain a high number of compositions to increase the sample size, hence improving the statistical power. Second, the subjects used robust model composition techniques and fragments of realistic design models. These improvements reduced "errors" that could obscure the causal relationship between the variable under study. Consequently, it brought a better reliability for our results.

5.2 Construct validity

It concerns the degree to which inferences are warranted from the observed cause and effect operations included in our study to the constructs that these instances might represent—i.e., "Are we actually measuring what we think we are measuring?" With this in mind, we evaluated (1) whether the quantification methods of the dependent variables are correct, (2) whether the quantification was accurately done, and (3) whether the different types of compositions can threat the validity.

Quantification methods The concept of effort used in our study is well known in the literature, and its quantification method was reused from previous work [22]. We quantified the dependent variables Effort(M_A, M_B), f(M_A, M_B), diff(M_{CM}, M_{AB}), and g(M_{CM}) based on the time (in minutes) invested by the subjects to perform each of them, while the correctness and inconsistency rate based on a suite of metrics, which was previously defined and independently validated in previous works [6,7]. We quantified the inconsistencies manually through several cycles of measurements and reviews, while the effort was recorded at the beginning and at the end of each experimental task.

The correctness of the quantification The authors have worked together to assure that the quantification of the variables was correctly performed. We checked whether the collected data were in line with the objective and hypotheses of our study. The quantification procedures were carefully planned and followed well-known quantification guidelines [13,15,16].

Execution of the compositions Another threat that we have controlled is if the use of manual or (semi)automated composition techniques might unintentionally influence the results. We have observed that the manual composition helps to minimize problems related to the composition tools. However, the use of the IBM RSA and Epsilon might jeopardize the results; as specific resources of the tools might influence the subjects during the execution of the experimental tasks. With this in mind, we have taken the precaution of making experimental decisions and seeking a study design that does not affect the results. First, the nature of the compositions did not require that the subjects understood the resources/details of the tools. Second, the size of the models and the complexity of the compositions were managed so that the use of these tools might not intentionally reduce (or exacerbate) the composition effort or even the generation of specific categories of inconsistencies in the output composed models. Therefore, we believe that the use of the model composition tool did not impose threats to the validity of our experimental results.

Finally, we have observed that the use of traditional, manual composition algorithms did not threat our findings. This affirmation is supported by some two reasons. First, even if the conflicting changes were manually identified or were unconsciously avoided, the algorithms were used as "rules of thumb" (guidelines), similarly in IBM RSA and Epsilon. Second, we have checked if the scope (and the size) of the compositions has influenced the accommodations of the changes from M_B to M_A using the traditional algorithms; if the number of model elements to be composed were high, then the manual composition might become so confusing. Thus, we kept it simple. The compositions were reviewed and observed during the experimental tasks.

5.3 Internal validity

Inferences between our independent variable (composition techniques) and the dependent variables (composition effort,

correctness and inconsistency rate) are internally valid if a causal relation involving these two variables is demonstrated [13,23,24]. Our study met the internal validity because: (1) the temporal precedence criterion was met, i.e., the composition of design models preceded the inconsistencies and composition effort; (2) the covariation was observed, i.e., the use of composition techniques led to varying accordingly to the composition effort; and (3) there is no clear extra cause for the detected covariation. Our study satisfied all these three requirements for internal validity.

We also analyzed the internal validity can be also supported by other means. First, we performed some cases for demonstrating how the dependent variables are being exclusively affected by the independent variable. Second, we have observed that the collected values for the inconsistency rates and composition effort were confidently caused by the change of the composition technique used.

However, some threats were also identified. First, as the measures of the dependent variables were partially calculated in a manual fashion, there was the risk that the collected data would not be always reliable. Hence, this could lead to inconsistent results. However, we have mitigated this risk by establishing measurement guidelines and two-round data reviews with the authors

Next, usually the confounding variable is seen as the major threat to the internal validity [24]. That is, rather than just the independent variable, an unknown third variable unexpectedly affects the dependent variable. Thus, a pilot study was carried out to make sure that the dependent variables were not affected by any existing variable other than the use of the composition techniques. During this pilot study, we tried to identify which other variables could affect the dependent variables such as the size of the models.

5.4 External validity

External validity refers to the validity of the obtained results in other broader contexts [25]. That is, to what extent the results of this controlled study can be generalized to other realities, for instance, with different composition techniques, design models with different sizes, with more experienced developers and quantifying other inconsistencies. Thus, we analyzed whether the causal relationships investigated could be held over variations in people, treatments, and other settings. As this study was not replicated yet, we made use of the theory of proximal similarity (proposed by Donald T. Campbell [20]) to identify the degree of generalization of the results. The goal is to define criteria that can be used to identify similar contexts where the results of this study can be applied.

Some criteria are shown as follows. First, the developers should be able to make use of composition techniques for evolving design models, i.e., UML class. Second, the composition should be implemented for evolving design models; more specifically, evolutions based on addition, exclusion, derivation, and change. It is important to highlight that the models used were small. Next, the developers should use one of the composition techniques investigated. Given that these criteria may happen in mainstream software development, we conclude that the results of our study may be generalized, at some point, to other contexts that are more similar to these requirements.

6 Related work

Model composition is a very active research field in many research areas such as merging of state charts [26], composition of software product lines [27], aspect-oriented models [28], and mainly composition of UML models [29–32]. Research initiatives tend to focus on proposing model composition techniques or even creating innovative modeling languages. However, the evaluation of the developers' effort on composing design models using the proposed techniques is still incipient. Hence, the lack of quantitative and qualitative indicators on composition effort may hinder mainly the understanding of side effects peculiar to certain composition techniques.

Some studies have notably aimed at evaluating modeling languages such as UML in terms of some quality attributes such as comprehensibility and completeness [4,33]. In [34], Petre investigated how exactly UML is being used in industry. For this, she performed 50 interviews with professional software engineers in 50 companies and identified 5 patterns of UML use. Although UML has been adopted, in fact, as the industry standard modeling language, it is just a point of investigation in empirical studies considering model composition. In general, most of the research on the interplay of effort and composition techniques rest on subjective assessment criteria of experts who have built up an arsenal of mentally held indicators to analyze the growing complexity of models and then evaluate the effort on composing them. Consequently, developers ultimately rely on feedback from experts to determine "how good" the input models and their compositions are. There are many composition techniques in the literature such as MATA [26], Epsilon [4], and IBM RSA [5]; but, they are rarely evaluated.

The literature on model composition has been formed largely by expert reflection and opinion rather than empirical evidence. There have been repeated calls for deeper investigation of actual effort that developers invest to integrate design models using these techniques in realistic settings [2]. Unfortunately, these approaches do not offer any empirical evidence or insights into the effort required to compose design models. As a matter of fact, the current literature about composition technique points out the absence of empirical

studies and does highlight the importance of empirical evidence [1,2,26,27]. According to [1], the state of the practice in assessing model quality provides evidence that modeling is still in the craftsmanship era and when we assess model composition this problem is accentuated. More specifically, to the best of our knowledge, our results are the first to empirically investigate the topics of the research questions in a controlled and systematic way using specification-based and heuristic-based techniques.

The current model composition literature does not evaluate the effect of the composition techniques on the effort to apply them, detect, and resolve the inconsistencies, as well as on the model correctness and inconsistency rate. In [1], the authors highlight the need empirical studies in model composition to provide insights about how deal with ever-present problems such as conflicts and inconsistencies in real-world settings. In [2], Mens also reveals the need of more "experimental researches on the validation and scalability of syntactic and semantic merge approaches, not only regarding conflict detection, but also regarding the amount of time and effort required to resolve the conflicts."

Some previous works investigated the effect of using UML diagrams and its profiles with different purposes. In [35], Briand et al. looked into the formality of UML models and its relation with model quality and comprehensibility. In particular, Briand and colleagues investigated the impact of using OCL (Object Constraint Language [36]) on defect detection, comprehension, and impact analysis of changes in UML models. In [12], Filippo et al. carried out a series of four experiments to assess how developer's experience and ability influence Web application comprehension tasks supported by UML stereotypes. Although they have found that the use of UML models provide real benefits for typical software engineering activities, none has investigated the peculiarities of UML models in the context of model composition. Finally, we see this paper as a first step in a more ambitious agenda to support empirically the assessment of model composition techniques in general.

7 Conclusions and future work

Model composition plays a pivotal role in many software engineering activities, e.g., evolving SPL design models to add new features and reconciling conflicting models developed in parallel. Many composition techniques have been proposed, including heuristic-based and specification-based techniques, for supporting the composition of these design models. This paper represents a first controlled experiment to assess the trade-off between the specification-based and heuristic-based techniques in terms of composition effort and model correctness. More specifically, we compare the impact of the techniques (IBM RSA, Epsilon, and traditional algorithms) on the effort to apply them, detect, and resolve inconsistencies, as well as their impact on the model correctness and inconsistency rate. The subject used the composition techniques to evolve design models based on six evolution scenarios.

Our initial hypotheses were that the specification-based techniques would reduce the composition effort and produce a higher number of correctly composed models than its counterpart. Surprisingly, we found that the specification-based techniques neither reduce the developers' effort nor guarantee the correctness of the compositions. Even worse, the traditional composition algorithms outnumber the specification-based technique to some extent. There are few studies assessing the effort required for the use of model composition techniques. Thus, this study can be seen as a first exploratory study that investigates the effect of composition techniques in a controlled manner.

Further empirical studies are still required to better understand whether these findings are confirmed or not in other contexts, considering other design models, encompassing different evolution scenarios, and evaluating other composition techniques. Although the techniques investigated are robust and representative and there are reasons to believe the results will possibly generalize to similar scenarios, we do not claim generalization beyond these techniques and their use in design models, in particular class diagrams. Finally, we hope that the issues outlined throughout the paper encourage other researchers to replicate our study in the future under different circumstances and that this work represents a first step in a more ambitious agenda on better supporting model composition tasks.

References

1. France, R., Rumpe, B.: Model-driven development of complex software: a research roadmap. In: Future of Software Engineering at ICSE'07, pp. 37–54 (2007)
2. Mens, T.: A state-of-the-art survey on software merging. IEEE Trans. Softw. Eng. **28**(5), 449–462 (2002)
3. Clarke, S.: Composition of object-oriented software design models. Ph.D. thesis, Dublin City University (2001)
4. Epsilon.: http://www.eclipse.org/gmt/epsilon/ (2013)
5. IBM Rational Software Architect. http://www.ibm.com/developerworks/rational/products/rsa/ (2013)
6. Farias, K.: Empirical evaluation of effort on composing design models. Ph.D. thesis, PUC-Rio, Rio de Janeiro, Brazil (2012)
7. Farias, K.: Empirical evaluation of effort on composing design models. In: 32nd ACM/IEEE International Conference on Software Engineering, Doctoral Symposium, vol. 2, pp. 405–408, Cape Town, South Africa (2010)
8. Clarke, S., Baniassad, E.: Aspect-Oriented Analysis and Design: The Theme Approach. Addison-Wesley, Upper Saddle River (2005)
9. Farias, K., Garcia, A., Whittle, J., Chavez, C., Lucena, C.: Evaluating the effort of composing design models: a controlled experiment, In: Proceedings of the 15th International Conference on Model-

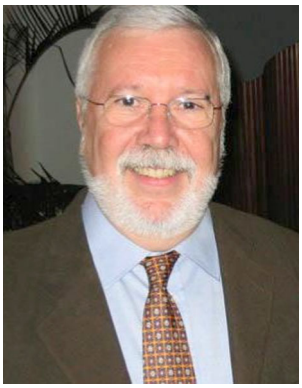
Driven Engineering Languages and Systems (MODELS'12), Innsbruck, Austria, vol. 7590, pp. 676–691 (2012)
10. Farias, K., Garcia, A., Whittle, J.: Assessing the impact of aspects on model composition effort. In: 9th International Conference on Aspect-Oriented Software Development (AOSD'12), Saint Malo, France, pp. 73–84 (2010)
11. Clarke, S., Walker, R.: Composition patterns: an approach to designing reusable aspects. In: 23rd International Conference on Software Engineering (ICSE'01), Toronto, ON, pp. 5–14 (2001)
12. Ricca, F., Penta, M., Torchiano, M., Tonella, P., Ceccato, M.: How developers' experience and ability influence web application comprehension tasks supported by UML stereotypes: a series of four experiments. IEEE Trans. Softw. Eng. **96**(1), 96–118 (2010)
13. Wohlin, C., et al.: Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers, Norwell (2000)
14. Cook, T., Campbell, D., Day, A.: Quasi-experimentation: Design and Analysis Issues for Field Settings. Houghton Mifflin, Boston (1979)
15. Kitchenham, B., Al-Kilidar, H., Babar, M., Berry, M., Cox, K., Keung, J., Kurniawati, F., Staples, M., Zhang, H., Zhu, L.: Evaluating guidelines for reporting empirical software engineering studies. Empir. Softw. Eng. **13**(1), 97–112 (2008)
16. Kitchenham, B.: Empirical paradigm: the role of experiments. In: Empirical Software Engineering Issues, pp. 25–32 (2006)
17. Sjøberg, D., Anda, B., Arisholm, E., Dybå, T., Jørgensen, M., Karahasanovic, A., Koren, E., Vokác, M.: Conducting realistic experiments in software engineering. In: 1st International Symposium on Empirical Software Engineering, pp. 17–26 (2002)
18. Basili, V., Caldiera, G., Rombach, H.: The Goal Question Metric Paradigm. Encyclopedia of Software Engineering. Wiley, New York (1994)
19. Devore, J., et al.: Applied Statistics for Engineers and Scientists. Duxbury, North Scituate (1999)
20. Campbell, D., Russo, M.: Social Experimentation. SAGE Classics, Beverly Hills (1998)
21. Research method knowledge base: improving conclusion validity. http://www.socialresearchmethods.net/kb/concimp.php (2011)
22. Jørgensen, M.: Practical guidelines for expert-judgment-based software effort estimation. IEEE Software, pp. 57–63 (2005)
23. Brewer, M.: Research design and issues of validity. In: Reis, H., Judd, C. (eds.) Handbook of Research Methods in Social and Personality Psychology. Cambridge University Press, Cambridge (2000)
24. Shadish, W., Cook, T., Campbell, D.: Experimental and Quasi-experimental Designs for Generalized Causal Inference. Houghton Mifflin, Boston (2002)
25. Mitchell, M., Jolley, J.: Research Design Explained, 4th edn. Harcourt, New York (2001)
26. Whittle, J., Jayaraman, P.: Synthesizing hierarchical state machines from expressive scenario descriptions. ACM Trans. Softw. Eng. Methodol. **19**(3), 1–45 (2010)
27. Thaker, S., Batory, D., Kitchin, D., Cook, W.: Safe composition of product lines. In: 6th Generative Programming: Concepts and Experiences, Salzburg, Austria, pp. 95–104 (2007)
28. Klein, J., Hélouët, L., Jézéquel, J., Semantic-based weaving of scenarios. In: 5th Aspect-Oriented Software Development (AOSD'06), Bonn, Germany, pp. 27–38 (2006)
29. Asklund, U.: Identifying inconsistencies during structural merge. In: Proceedings of Nordic Workshop Programming Environment Research, pp. 86–96 (1994)
30. Farias, K., Garcia, A., Lucena, C.: Effects of stability on model composition effort: an exploratory study. J. Softw. Syst. Model. (SoSym) **12**(1), 1–22 (2013)
31. Norris, N., Letkeman, K.: Governing and managing enterprise models: part 1. Introduction and concepts. IBM Developer Works, www.ibm.com/developerworks/rational/library/09/0113_letkeman-norris (2011)
32. Perry, D., Siya, P., Votta, L.: Parallel changes in large scale software development: an observational case study. In: International Conference on Software Engineering (ICSE'98), pp. 251–260 (1998)
33. Lange, C., Chaudron, M.: Effects of defects in UML models: an experimental investigation. In: International Conference on Software Engineering (ICSE'06), China, pp. 401–410 (2006)
34. Petre, M.: UML in practice. In: 35th International Conference on Software Engineering (ICSE 2013), San Francisco, CA, pp. 18–26 (2013)
35. Briand, L., Labiche, Y., Di Penta, M., BondocL, H.: An experimental investigation of formality in UML-based development. IEEE Trans. Softw. Eng. **31**(10), 833–849 (2005)
36. Unified Modeling Language: Infrastructure, Object Management Group (2010)

Kleinner Farias is an Assistant Professor in the Interdisciplinary Postgraduate Program in Applied Computing at the University of Vale dos Rio dos Sinos (Unisinos). He is an associate member of the OPUS Researcher Group at the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil. He received his Ph.D. in Computer Science from PUC-Rio in 2012. He received his Masters degree in Computer Science from the Pontifical Catholic University of Rio Grande do Sul in 2008. He completed his undergraduate studies in Computer Science at the Federal University of Alagoas and in Information Technology at the Federal Institute of Alagoas in 2006. His current research interests include software modeling, empirical evaluation of model composition techniques, model-driven software development, software metrics and software product lines.

Alessandro Garcia is an Assistant Professor in the Informatics Department at PUC-Rio, Rio de Janeiro, Brazil. He was previously a Lecturer in the Computing Department at Lancaster University (United Kingdom), from 2005 to 2009. His research interests embrace several topics in the broad area of software engineering, including software architecture, modularity, software metrics, error handling, and software product lines. He has been serving as a Program Committee member of premier international conferences on software engineering, such as ICSE, FSE, AOSD/Modularity, MODELS, ICPC, ESEM, SPLC, and many others. He received many awards, distinctions, and recognitions, including Best Dissertation Award (Brazilian Computer Society, 2000), Best Researcher Award (Lancaster University, 2006), Distinguished Young Scholar (PUC-Rio, 2009 and 2012), and Young Scientist Fellowship (FAPERJ, 2009 and 2013). He holds a CNPq productivity grant (level 1D) and is an affiliate member of the Brazilian Academy of Sciences (ABC).

Jon Whittle is full Professor and Chair of Software Engineering at Lancaster University. He is also a Royal Society Wolfson Merit Award Scholar. He has been working with Model-Driven Engineering for over ten years, including stints at NASA Ames Research Center (California), George Mason University (Virginia), IIT Kanpur (India), and more recently at Lancaster University. He has been intimately involved with the MDE research community during this time, serving as the Chair of the Steering Committee of the MODELS conference from 2006 to 2008 and as PC Chair in New Zealand in 2011. He also serves on the editorial board of the Journal of Software and System Modeling. Jon's current interests are in empirically investigating what factors lead to success or failure with MDE in industry.

Christina von Flach Garcia Chavez is an Associate Professor at the Department of Computer Science (DCC) of the Federal University of Bahia (UFBA) since 1990. She has Ph.D. in Computer Science from the Pontifical Catholic University of Rio de Janeiro (2004), Master in Computer Science from the State University of Campinas (1992), and B.S. in Computer Science from the Federal University of Bahia (1987). She is currently coordinating the Ph.D. Graduate Program in Computer Science (PMCC) at UFBA. She participates in the Software Engineering Laboratory of UFBA (LES @ UFBA) and coordinates the research group on Software Design and Evolution at UFBA (aside @ UFBA). Her research interests include various topics related to software architecture and evolution, and software engineering education.



Carlos Lucena is a Full Professor of Computer Science at the Pontifical Catholic University of Rio de Janeiro (PUC-Rio) since 1982 and an Adjunct Professor of Computer Science and a Senior Research Associate of the Computer Systems Group at the University of Waterloo, which he has visited on a regular basis since 1975. He completed his undergraduate studies in Economics and Mathematics between 1962 and 1965 at PUC-Rio and received his Masters degree from the University of Waterloo (1969), Canada, and his Ph.D. from the University of California in Los Angeles (1974). His current research focuses on agent-oriented software engineering, multi-agent applications, autonomic computing and software reuse.