Evaluating the Effort of Composing Design Models: A Controlled Experiment

Kleinner Farias[1], Alessandro Garcia[1], Jon Whittle[2], Christina Chavez[3], and Carlos Lucena[1]

[1] OPUS Research Group/LES, Informatics Department, PUC-Rio, Brazil
{kfarias,afgarcia,lucena}@inf.puc-rio.br
[2] School of Computing and Communications, Lancaster University, UK
whittle@comp.lancs.ac.uk
[3] Department of Computer Science, Federal University of Bahia, Brazil
flach@dcc.ufba.br

Abstract. The lack of empirical knowledge about the effects of model composition techniques on developers' effort is the key impairment for their widespread adoption in practice. This problem applies to both existing categories of model composition techniques, i.e. specification-based (e.g. Epsilon) and heuristic-based (e.g. IBM RSA) techniques. This paper reports on a controlled experiment that investigates the effort to: (1) apply both categories of model composition techniques, and (2) detect and resolve inconsistencies in the output composed models. The techniques are investigated in 144 evolution scenarios, where 2304 compositions of elements of class diagrams were produced. The results suggest that: (1) the employed heuristic-based techniques require less effort to produce the intended model than the chosen specification-based technique, (2) the correctness of the output composed models generated by the techniques is not significantly different, and (3) the use of manual heuristics for model composition outperforms their automated counterparts.

Keywords: Model composition effort, empirical studies, effort measurement.

1 Introduction

Model composition plays a central role in many software engineering activities, including the evolution of design models [5,8]. Developers may spend some considerable effort applying model composition techniques to compose M_A and M_B. As a consequence, both academia and industry are increasingly concerned with developing effective techniques for composing design models (e.g. [5,10][14-19]). 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, in order to produce an output intended model, M_{AB}.

Existing techniques that support model composition can be classified as specification-based techniques (e.g. Epsilon [15]), and heuristic-based techniques (e.g. the heuristics supported by the IBM Rational Software Architect (RSA) [16]). In the first case, developers explicitly specify the correspondence and composition relations

R.B. France et al. (Eds.): MODELS 2012, LNCS 7590, pp. 676–691, 2012.

between the elements of the input models (M_A and M_B) to give rise to M_{AB}. In the second case, developers use a set of predefined heuristics, which "guess" the relations between the elements of M_A and M_B before producing M_{AB}.

However, instead of producing the output intended model, M_{AB}, as would be expected, the technoiques may produce an output composed model, M_{CM}, with inconsistencies. These inconsistencies often result from the incorrect resolution of conflicting changes between the model element from M_A and M_B. If M_{CM} and M_{AB} do not match ($M_{CM} \neq M_{AB}$) due to inconsistencies in M_{CM}, developers will need to invest some extra effort to detect and resolve the inconsistencies in M_{CM} so that it can be transformed into M_{AB}. Note that the key motivation for applying composition techniques is to reduce the effort of the developers to produce the output intended model [17, 18]. The proponents of specification-based techniques claim that explicit composition specifications entail a more systematic way to compose M_A and M_B [8, 17]; hence, developers expect to save effort by using them. That is, the conventional wisdom [5, 8, 17, 18] is that a precise composition specification favors the production of correctly composed models (i.e. where $M_{CM} = M_{AB}$), thereby minimizing the developers' effort.

To date, however, there is little evidence to confirm (or not) this expectation. As a result, developers use model composition techniques without any support of empirical knowledge regarding their effects on the effort to apply them as well as to detect and resolve inconsistencies in M_{CM}. If a particular composition technique reduces effort, but has a detrimental effect on the model correctness (or vice-versa), it is quite arguable whether developers may use it in mainstream software projects, where time and cost are tight. Having empirical knowledge at hand, developers can choose and adopt composition techniques in a rational way. Today, the adoption of the techniques is based on evangelists (often divergent) opinions.

This paper reports empirical findings on the use of specification-based and heuristic-based composition techniques (Section 2.3) to evolve design models. We have conducted a controlled experiment to evaluate and compare such techniques with respect to the developer's effort and model correctness in the context of evolving design models (Section 3). A total of 24 subjects carried out 144 compositions of UML class diagrams with the support of such techniques. The comparative analysis (Section 4) embodied the effort of applying alternative composition techniques, detecting inconsistencies and resolving them in the output composed model. The main surprising results, supported by statistical analysis, suggest that: (1) the specification-based technique required more 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 assessed techniques.

The contributions of this paper are (a) empirical findings on the impact of heuristic and specification-based composition techniques on developers' effort to apply techniques, detect inconsistencies and resolve inconsistencies; (b) insights about how to evaluate the developers' effort, reduce error proneness in model composition, and minimize side effects of composition techniques in practice; and finally, (c) to serve as an in-depth example of how controlled experiments can be conducted to evaluate and compare model composition techniques. We also discuss the threats to validity (Section 5), the limitations of related work (Section 6), and concluding remarks (Section 7). Although we cannot generalize our empirical findings to other model composition

techniques, our exploratory experiment stands for a trailblazing contribution to improve understanding of the potential effects of model composition techniques on developers' effort.

2 Background

2.1 Model Composition Effort

In this study, model composition effort is described by an effort equation (Fig. 1). Developers often invest effort to realize three activities to compose the base model, M_A, (the model to-be changed) and the delta model, M_B (i.e. the changes), to produce M_{CM}. The first activity, the application of the model composition technique, is represented by $f(M_A,M_B)$ in the equation. The additional effort is usually invested to detect inconsistencies in M_{CM} – represented by, $diff(M_{CM},M_{AB})$ – and to resolve these inconsistencies – represented by $g(M_{CM})$. If M_{CM} perfectly matches the intended model, M_{AB}, then $diff(M_{CM}, M_{AB}) = 0$ and $g(M_{CM}) = 0$. Otherwise, additional effort is required to deal with inconsistencies, meaning that $diff(M_{CM},M_{AB}) > 0$ and $g(M_{CM}) > 0$.

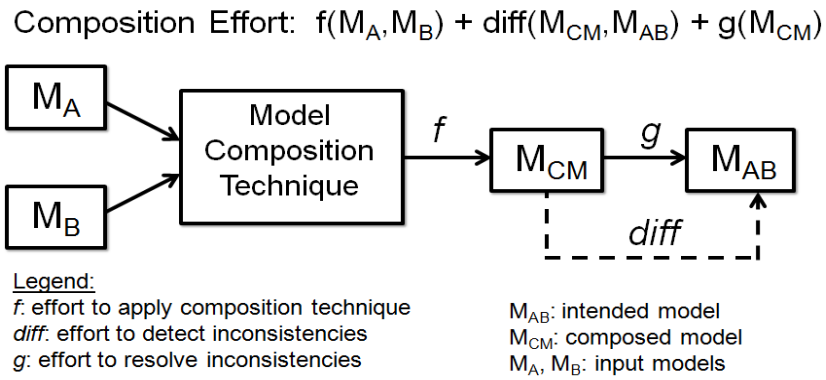

Fig. 1. Overview of model composition effort: an equation.

2.2 Composition Conflicts and Inconsistencies

It is well known that the properties of the model elements of M_A and M_B may conflict with each other. Fig. 2 shows a simple example of composition conflict. In the base model, the *Researcher* is defined as a concrete UML class (i.e. *Researcher.isAbstract* = false) whereas in the delta model, *Researcher* is an abstract class (i.e. *Researcher.isAbstract* = true). Before composing, the developers need to properly answer the question: should class *Researcher* be an abstract class (or not)? In this particular case, the correct answer is that the *Researcher* must be abstract (see the intended model in Fig. 2). However, conflicts may be converted into inconsistencies in M_{CM} when unexpected values are set to the properties of the model elements. Fig. 2 shows that the class *Researcher* produced by the override and merge algorithms (Section 2.3) is a concrete class (*isAbstract* = false) instead of an abstract one (*isAbstract* = true), as would be expected. Because of this inconsistency, the output composed model is *not compliant* with the intended model. Two categories of inconsistencies can emerge, including:

- *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 intended model elements. For instance, a class in M_{CM} has an unexpected method, or it requires functionality from other classes that no longer exist after the composition.

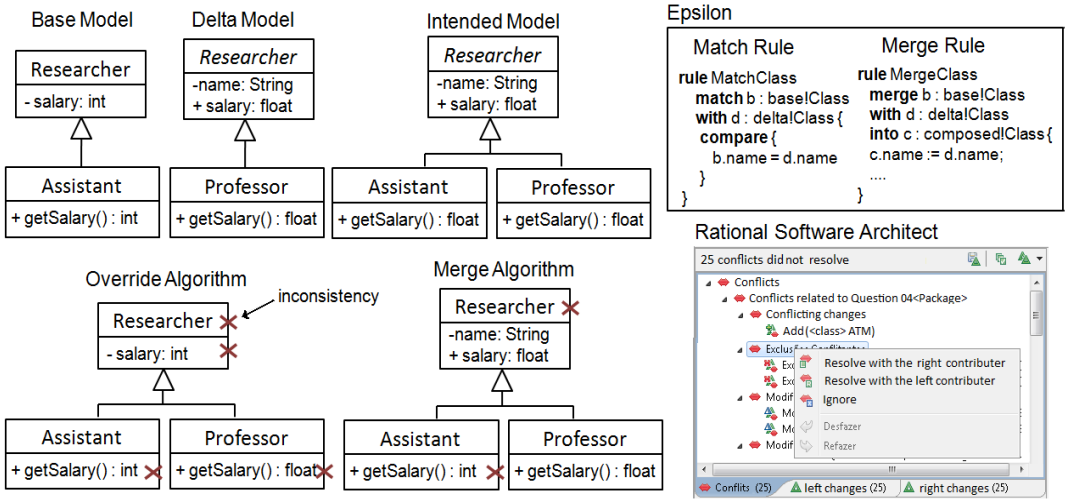

Fig. 2. Illustrative example.

In our study, we focus on semantic inconsistencies because they cannot be automatically identified using model composition techniques. They often require some intervention from software developers. In addition, they are mainly responsible for non-trivial composition problems during the model evolution [8]. As a consequence, they often require more effort and are more detrimental to the correctness of the output model than syntactic inconsistencies [9]. The categories of semantic inconsistencies considered are: (1) a model element in M_{CM} is not compliant with the corresponding one in M_{AB}; (2) model elements are missing from M_{CM}, or should nor be defined in M_{CM}; (3) model elements are unexpectedly duplicated according to 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 are the most common types of problems faced by developers dealing with model inconsistencies [4,8]. In our study, we explicitly discriminate each contradicting change (i.e. the conflict) from its consequence in the output model (i.e. the inconsistency).

2.3 Model Composition Techniques

The composition techniques used in our study were Epsilon [15], the representative of specification-based techniques, and two representatives of heuristic-based techniques, namely the IBM RSA [16] and traditional composition algorithms (TRA) [9]. These three techniques were selected as they provide different degrees of automation support. The selected heuristic-based techniques include both an automated technique

(RSA) and a manual technique (TCA) to support model composition. Specification-based techniques cannot be applied manually. Epsilon and IBM RSA are supported by robust, usable tools, an essential prerequisite for a controlled experiment like ours. IBM RSA is an industry-leading tool and it is the most widely used tool in the industry [16]. Epsilon is stable, easy-to-use tool for specification-based composition that was available for our study. 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 [10, 19]. These techniques are described as follows.

Epsilon (EPS). It provides a hybrid, rule-based language for merging design models [15]. Developers invest effort to edit a set of match and merge rules before producing M_{AB}. Fig. 2 shows an example of these rules. The merge rule specifies that all classes to be composed will have the names of classes from 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 [16]. IBM RSA is characterized as a semi-automated model composition technique. Like the Epsilon technique, its use does not ensure that M_{AB} will be always produced. By using the IBM RSA developers should interactively resolve conflicts before producing M_{AB}. Fig. 2 depicts an example of a conflict report. When conflicting changes emerge, developers should decide which changes will be inserted into the output composed model — from the base model (*Researcher.isAbstract* = false) or from the delta model (*Researcher.isAbstract* = true).

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 [9]. These algorithms were chosen for several reasons. First, model evolution scenarios can be decomposed into one or more operations supported by a combination of these algorithms. Second, these algorithms are often used as guidelines for the developers composing OO models manually [10, 19]. Third, we wanted to investigate to what extent the aforementioned automated techniques outperform the use of a classical manual technique for model composition. In the following, we provide a brief definition for override and merge algorithms to be applied to 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. 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 involved in a correspondence matching 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 Experiment Planning

3.1 Experiment Definition

The objective of this study is stated based on the GQM template 2 as follows:

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

Based on this, we focus on the two research questions:

RQ1: What is the relative effort of composing two input models by 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 techniques than heuristic-based techniques?

3.2 Hypothesis Formulation

Hypothesis 1. We conjecture that although specification-based composition techniques provide a more systematic way to compose the input models, they do not reduce the overall composition effort in practice. We suspect that developers have to invest too much effort to specify the compositions; but, this additional effort is not converted into a higher number of correctly composed models than that produced with heuristic techniques. However, it is by no means obvious that this hypothesis holds. It may be, for example, that specification-based techniques help developers to match and then compose the input models more quickly.

Null Hypothesis 1, $H_{1\text{-}0}$: The specification-based composition technique requires less (or equal) effort than the heuristic-based ones to produce M_{AB} from M_A and M_B.

$H_{1\text{-}0}$: $\text{Effort}(M_A,M_B)_{\text{Specification}} \leq \text{Effort}(M_A,M_B)_{\text{Heuristic}}$

Alternative Hypothesis 1, $H_{1\text{-}1}$: The specification-based technique requires more effort than the heuristic-based ones to produce M_{AB} from M_A and M_B.

$H_{1\text{-}1}$: $\text{Effort}(M_A,M_B)_{\text{Specification}} > \text{Effort}(M_A,M_B)_{\text{Heuristic}}$

We refine this hypothesis in other three *subhypotheses* ($H1_2$, $H1_3$, and $H1_4$). A formulation for these hypotheses is presented in Table 1.

Hypothesis 2. The specification-based technique is expected to produce a higher number of correctly composed models as developers can precisely express the composition relations between the input models. However, it is not clear whether this composition technique can, in fact, help developers to improve the correctness (Cor) of the output model when compared to the use of heuristic approaches. These hypotheses are presented as follows:

Null Hypothesis 2, $H_{2\text{-}0}$: The specification-based technique produces a lower (or equal) number of correctly composed models than the heuristic-based techniques.

$H_{2\text{-}0}$: $\text{Cor}(M_{CM})_{\text{Specification}} \leq \text{Cor}(M_{CM})_{\text{Heuristic}}$

Alternative Hypothesis 2, $H_{2\text{-}1}$: The specification-based technique produces a higher number of correctly composed models than the heuristic-based technique.

$H_{2\text{-}1}$: $Cor(M_{CM})_{Specification} > Cor(M_{CM})_{Heuristics}$

The composition correctness is influenced by the presence (or not) of inconsistencies in the output composed model. Thus, we investigate if the specification-based technique entails (or not) a lower inconsistency rate than the use of the heuristic-based techniques. This new elaborated hypothesis is stated in Table 1.

Table 1. Tested hypotheses

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

Effort: Effort to compose the input models (RQ1), S: Specification-based composition technique.

f: Effort to apply the composition techniques (RQ1), H: Heuristic-based.

diff: Effort to detect inconsistencies (RQ1), g: Effort to resolve the inconsistencies (RQ1).

Cor: Correctness of the composition (RQ2), Rate: Inconsistency rate of the composed model (RQ2).

3.3 Context and Subject Selection

The subjects used the Epsilon, IBM RSA and the traditional algorithms to produce model compositions for six software evolution scenarios (Table 2). None of the subjects were familiar beforehand with either the design models or the required changes. The selected evolution scenarios were tasks where developers are not the initial designers of the models. The design models used were fragments of industrial models captured from different application domains, such as financial and simulation of petrol extraction. The experiment was conducted with 16 subjects were professionals from Brazilian companies and 8 subjects were students with professional experience [19]. All professionals held a Master's degree, Bachelor's degree or equivalent, and had a considerable knowledge of software modeling and programming to participate in the experiment [19]. The students were also invited to participate in the experiment, so that we could have subjects with different backgrounds and levels of expertise [1]. They were from two Master and Doctoral programs in Computer Science at two Brazilian universities: Pontifical Catholic University of Rio de Janeiro (PUC-Rio) and the Federal University of Bahia (UFBA). These students attended either a course on "empirical studies in software engineering" at PUC-Rio or a course on "software evolution" at UFBA. The experiments were part of the courses and were performed as practical laboratory exercises. The participant was exposed to the same level of training on the model composition techniques under assessment [19].

Table 2. The 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.

3.4 Experimental Design

The experimental design of this study is characterized as a *randomized complete block one* with three treatments, i.e. the use of the three composition techniques. The study had a set of activities that were organized into 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 [1]. 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*. Fig. 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 upon a description of changes (Table 2) that defines how the model elements of M_A were changed. Note that M_B, the delta model, was pre-prepared. 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 (Section 4.3). It is important to point out that a participant (subject x) produced M_{CM} in the first phase; in the second phase, other participant (subject n-x) detected and resolved the inconsistencies in M_{CM} in order to produce M_{AB}.

Detect inconsistencies. Subjects reviewed M_{CM} to detect inconsistencies. To this end, they checked if 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.

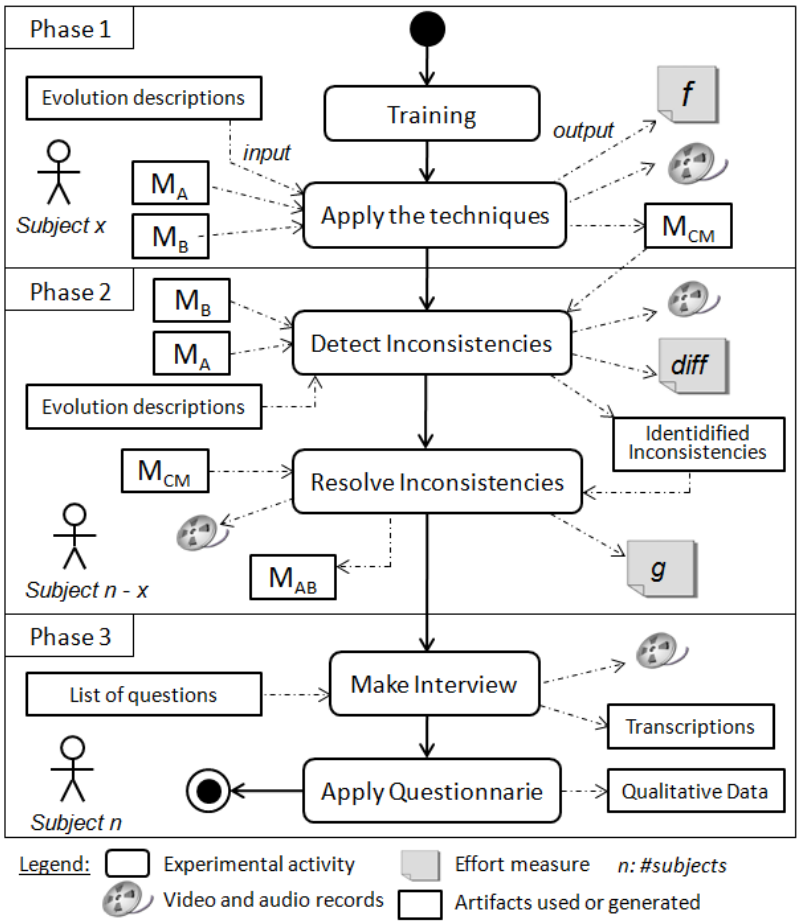

Fig. 3. The experimental process

Resolve inconsistencies. The subjects resolved the inconsistencies previously localized in order 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 academic background and work experience and apply some inquisitive questions.

Material. The models used in our study were UML class diagrams with 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 [20]. For example, Asklund et al. [18] 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.

Variables. The independent variable of this study is the choice of composition techniques. We investigate the impact of this independent variable in the following dependent variables:

- Effort. This variable measures the overall time (in minutes) invested by subjects to compose the input models ($H_{1\text{-}1}$). It is elaborated in three other variables: effort to apply model compositions ($H_{1\text{-}2}$), effort to detect inconsistencies ($H_{1\text{-}3}$), and effort to resolve inconsistency ($H_{1\text{-}4}$).
- Correctness. The full correctness of a composition ($H_{2\text{-}1}$) is ensured when the output composed model produced is *correct* with respect to the description of the intended change request (i.e. $M_{CM} = M_{AB}$). We have compared the produced models with the intended models (our 'reference intended models'), produced by the actual developers of those systems from where the input models were extracted. The composed model produced may be rated as either *correct* or *incorrect.* Note that a composed model with one of the previously described inconsistencies (Section 2.2) would be deemed as *incorrect.* We also investigate the *inconsistency rate* of the incorrectly composed model. It represents the ratio of the number *of inco*nsis*tencies* of a composed model divided by its number of model elements ($H_{2\text{-}2}$). The actual developers were consulted when we were unsure about particular inconsistencies in the composed models produced by the subjects.

4 Experimental Results

4.1 RQ1: Effort and Composition Techniques

Descriptive Statistics. The developers invest less effort to produce M_{AB} by using heuristic-based techniques rather than the specification-based technique. In fact, they spent less effort to apply the composition techniques (*f*), detect inconsistencies (*diff*), and resolve inconsistencies (*g*) (Table 3). The traditional algorithms required less effort than the IBM RSA, which in turn required less than the Epsilon. This is a very interesting finding because the common sense would be otherwise i.e., developers would invest less effort by using the Epsilon and IBM RSA. Table 3 shows the descriptive statistics of the collected data. Regarding the median of the general effort, it grew significantly from 11 to 14 and 21 by using RSA and Epsilon, respectively. This superior effort represents an increase by about 27.27 and 90.90 percent. This upward trend was also observed in *f, diff*, and *g*. This evidence, therefore, demonstrates that the developers, in fact, tend to invest less effort with heuristic-based techniques than specification-based one.

Hypothesis Testing. Since the Shapiro-Wilk test [1] 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 checking if there exist significant differences among the three techniques under investigation. We test H1 (and its subhypotheses) 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 (and *f*, *diff* and *g*) using heuristic-based techniques were significantly lower than using automated techniques in all cases. Still by using the traditional algorithms this significance is higher. Thus, we can reject the H1 null hypotheses (and its $H1_{1\text{-}0}$, $H1_{2\text{-}0}$, $H1_{3\text{-}0}$ e $H1_{4\text{-}0}$). For example, in row 1 of Table 4, for measure *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 by using the IBM RSA is significantly lower than one using Epsilon. From row 1 it is also possible to notice that only one null hypothesis was not rejected, and 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 is confirmed.

Table 3. Descriptive statistic for the composition effort

	Effort			**f**			**diff**			**g**		
	[1] RA	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
St.D.	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, StD: Standard Deviation, TRA: traditional, RSA: Rational Software Architect, EPS: Epsilon.

Table 4. Wilcoxon test results for the composition effort

		General Effort			**f(M_A,M_B)**			**diff(M_{CM},M_{AB})**			**g(M_{CM})**		
task	**S**	**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 vs RSA, B: TRA vs EPS, C:RSA vs EPS, *p: p-value,* S*: statistics.*

4.2 RQ2: Correctness and Composition Techniques

Descriptive Statistics. Fig. 4 shows the correctness of the compositions generated by using the three techniques: traditional algorithms, Epsilon, and IBM RSA in six experimental tasks. The y-axis represents the proportions of the number of M_{AB} achieved by the number of compositions realized in each task using each composition technique, while the x-axis consists of the experiment 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 proportion of correctly composed models than RSA and EPS. That is, the intended model was generated in 42.86 percent of the cases in TRA, whereas 57.14 percent 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 71.43 percent of the cases, while EPS and RSA produced 28.57 and 57.14 percent of the cases, respectively.

Although TRA has obtained low measures in task 3 in comparison to task 2 (a decrease from 71.43 to 42.86 percent), it still got a superior value compared to EPS and RSA, i.e. value by about three times higher than the measure of EPS and RSA, comparing 42.86 and 14.29 percent. On the other hand, in task 6, this superiority was reversed. RSA got double the value than TRA and EPS, comparing 28.57 and 57.14 percent. Still subjects obtained the intended model by 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 61.90 percent of the compositions using TRA against 59.52 and 42.86 percent using the RSA and Epsilon technique, respectively.

Table 4 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 technique in most cases. This means that developers will not produce correctly composed model by using a technique based on composition specifications. Rather, the output models will have equal (or even more) inconsistency rate.

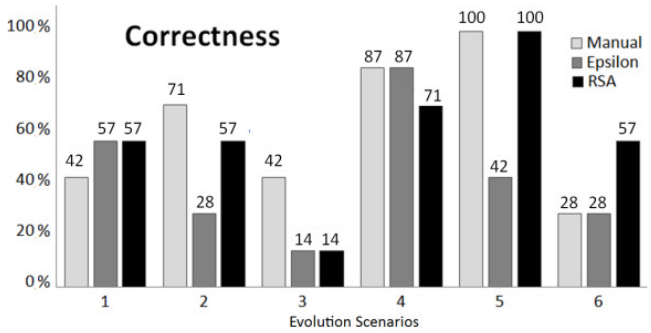

Fig. 4. The correctness of the output composed model

Table 5. The descriptive statistics for the inconsistancy rate

	N	Med	75th	Max	Mean	St D.
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

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 percent, respectively. Still note that the inconsistence rate in RSA is also higher than in TRA. In short, the inconsistency rate in EPS is higher than RSA, which outnumber TRA. This suggests that the inconsistency rate have favored TRA in comparison with RSA and 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. We apply the McNemar test to test $H2_1$. Table 6 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 $H2_{1\text{-}0}$ 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 the correctly composed models of the composition techniques.

We test $H2_2$ by applying the Wilcoxon test. Table 7 depicts the pairwise p-values for each measure. Bold p-values point out 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 the p-value is lower than 0.05 ($p = 0.0301$) for the measure between TRA vs EPS. This means that the inconsistency 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 vs RSA. This means that the inconsistency rate for TRA is lower, but no significantly lower than RSA.

Table 6. The descriptive statistic for the inconsistancy rate

Task	Comparison	χ^2	p-value
all	TRA vs RSA	0.27	0.606
	TRA vs EPS	0.75	0.387
	RSA vs EPS	0	1

Table 7. The descriptive statistic for the inconsistancy rate

Tasks	*Statistic*	Inconsistency Rate		
		tra vs rsa	*tra vs eps*	*eps vs rsa*
All	p-value	0.4851	**0.0301**	**0.0011**
	W	-5	**250**	**344**

W: sum of signed ranks.

4.3 Additional Observations

We have analyzed the qualitative data (i.e. interviews, video and audio records) to try explaining the results previously mentioned. First, the subjects mentioned that they often had some additional difficulties to match and compose the input model elements by using the specification-based composition techniques. Since they had difficulties to express the semantics of the changes required in each evolution scenario, given the problem at hand. This problem was observed in compositions dominated by relations between the input model elements of the type one-to-many (1:N) or many-to-many (N:N). The following extract from the interview also illustrates, for example, the difficulty related to the understanding of the scope of elements involved to specify a

composition: "*...express the changes in match and merge rules is boring...because all overlapping parts of the two input models should be analyzed...this is not a trivial task.*" Second, the IBM RSA tool shows the commonalities and differences between the input models in multiple, partial views. This strategy jeopardizes the creation of a "big picture view" of the output intended model. 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 happen... because I have to mentally "infer" a complete, unique view. On the other hand, the "strict" uses of the traditional algorithms are much more intuitive and allow me to freely work closer to the manner that I think that about model composition is.*"

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 the model elements; (2) the techniques should represent the conflicts between the input models in more innovative views; and (3) new composition techniques should be a mixture of specification-based and heuristic-based techniques. As a possible follow-up work, we would suggest to design intelligent recommendation systems that help developers to indicate what the best model composition strategy to-be applied, or even recommending how the input models should be restructured to save effort, whereas preventing inconsistencies. Moreover, the future techniques might provide "richer" visualization means to help developers to prevent inconsistencies before model compositions happen. Instead of merely reporting conflicting changes and inconsistencies, the techniques might provide layers and visualization filters of both conflicting changes and inconsistencies. Thus, developers could intuitively identify how the input model elements conflict with each other and how the inconsistencies propagate through the elements of the output composed model.

5 Threats to Validity

Statistical Conclusion Validity. Experimental guidelines were followed to eliminate this threat [2]: (1) the assumptions of the statistical tests (paired t-test and Wilcoxon) were not violated; (2) collected datasets were normally distributed; (3) the homogeneity of the subjects' background was assured; (4) the method of quantification was properly applied; and (5) statistical methods were used. The Kolmogorov-Smirnov and Shapiro-Wilk tests [2] were used to check how likely the collected sample was normally distributed. *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. That is, it answers the question: "Are we actually measuring what we think we are measuring?" All variables of this study were quantified based on a previous study [4]. Thus, they were defined and independently validated. Moreover, the concept of effort used in our study is well known in the literature [10]. Therefore, we are sure that the quantification method used is correct, and the quantification was accurately done. *External Validity*. We analyzed whether the causal relationships investigated during this study could be held over variations in people, treatments, composition techniques and the design models. There

are reasons to believe the results generalize beyond the three techniques used, but leave it to further work to fully test this.

6 Related Work

Model composition is a very active research field in many research areas such as merging of state charts [7], composition of software product lines [11], aspect-oriented models [12] and mainly UML models. 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. The lack of quantitative and qualitative indicators on composition effort hinders mainly the understanding of side effects peculiar to certain composition techniques.

Current work has notably aimed at evaluating modeling languages such as UML in terms of some quality attributes such as comprehensibility [14], completeness. 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 [5]. Even worse, this leads to dependence on 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 [4]. Consequently, the truth is that developers ultimately rely on feedback from experts to determine "how good" the input models and their compositions are. There are many examples in the literature of composition techniques such as MATA [7], Epsilon [15], and IBM RSA [16]. But, they will only be useful if the quality of the output composed models (e.g. correctness) is assured, and the composition effort required is low. Unfortunately, these approaches do not offer any insight or empirical evidence about the effort required to compose design models. As a matter of fact, the current literature about the composition technique points out the absence of empirical studies and does highlight the importance of empirical evidence [5,7,8,12].

According to [5], 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 way and systematic by using specification-based and heuristic-based techniques.

7 Concluding Remarks and Future Work

This paper can be seen as a first step to systematically assess the trade-off between the specification-based and heuristic-based techniques in terms of effort and correctness. The results of this first controlled experiment suggested that the specification-based techniques neither reduce the developers' effort nor guarantee the higher number of

correctly composed models. Even worse, the traditional composition algorithms outnumbered the specification-based technique, to some extent.

However, further empirical studies are still required to investigate if our results can be 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 other similar scenarios, we do not claim this generalization beyond these techniques, and their use applied to the design models, in particular, class diagrams. Finally, we expect that our findings can be used to motivate other studies.

References

1. Wohlin, et al.: Experimentation in Software Engineering: an Introduction. Kluwer Academic Publishers, Norwell (2000)
2. Devore, J., et al.: Applied Statistics for Engineers and Scientists. Duxbury (1999)
3. Basili, V., Caldiera, G., Rombach, H.: The Goal Question Metric Paradigm. In: Encyclopedia of Software Engineering, vol. 2, pp. 528–532. John Wiley and Sons (1994)
4. Farias, K., Garcia, A., Whittle, J.: Assessing the Impact of Aspects on Model Composition Effort. In: AOSD 2012, Saint Malo, France, pp. 73–84 (2010)
5. France, R., Rumpe, B.: Model-Driven Development of Complex Software: A Research Roadmap. In: Future of Software Engineering at ICSE 2007, pp. 37–54 (2007)
6. Unified Modeling Language: Infrastructure, Object Management Group (February 2010)
7. Whittle, J., Jayaraman, P.: Synthesizing Hierarchical State Machines from Expressive Scenario Descriptions. ACM TOSEM 19(3) (January 2010)
8. Mens, T.: A State-of-the-Art Survey on Software Merging. IEEE Trans. on Soft. Engineering 28(5), 449–462 (2002)
9. Clarke, S.: Composition of Object-Oriented Software Design Models, PhD thesis, Dublin City University (2001)
10. Jørgensen, M.: Practical Guidelines for Expert-Judgment-Based Software Effort Estimation. IEEE Software, 57–63 (May 2005)
11. Thaker, S., Batory, D., Kitchin, D., Cook, W.: Safe Composition of Product Lines. In: 6th GPCE, Salzburg, Austria, pp. 95–104 (2007)
12. Klein, J., Hélouët, L., Jézéquel, J.: Semantic-based Weaving of Scenarios. In: 5th AOSD 2006, Bonn, Germany, pp. 27–38 (March 2006)
13. Dingel, J., Diskin, Z., Zito, A.: Understanding and Improving UML Package Merge. Journal of Soft. and Syst. Modeling 7(4), 443–467 (2008)
14. Lange, C., Chaudron, M.: Effects of Defects in UML Models – An Experimental Investigation. In: ICSE 2006, China, pp. 401–410 (2006)
15. Epsilon (2011), `http://www.eclipse.org/gmt/epsilon/`
16. IBM RSA (2011), `http://www.ibm.com/developerworks/rational/products/rsa/`
17. Clarke, S., Baniassad, E.: Aspect-Oriented Analysis and Design: The Theme Approach. Addison-Wesley, Upper Saddle River (2005)
18. Asklund, U.: Identifying Conflicts during Structural Merge. In: Proc. Nordic Workshop Programming Environment Research, pp. 231–242 (1994)
19. Evaluating the Effort of Composing Design Models: A Controlled Experiment (2012), `http://www.les.inf.puc-rio.br/opus/models12-app`