Evaluating the Effects of Stability on Model Composition Effort: an Exploratory Study

Kleinner Farias, Alessandro Garcia, Carlos Lucena

[1]OPUS Research Group – LES – Informatics Department
Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
Rio de Janeiro – RJ– Brazil

{kfarias,afgarcia,lucena}@inf.puc-rio.br

Abstract. *Heuristics are often used to support model composition, but they also lead to syntactic and semantic inconsistencies in the composed models. If the effort to resolve inconsistencies is high, heuristic model composition might become impractical. Unfortunately, little is known whether well-designed models can minimize the inconsistency rate so that state-of-practice heuristics can be efficiently applied. This paper presents an exploratory study that analyzes the impact of model stability on the composition effort required to produce several releases of a software product line. Our results, supported by statistical tests, show that when models are well-structured upfront and, therefore more stable over time, the inconsistency rate and inconsistency resolution effort are kept under control. On the other hand, when changes are not predicted upfront, the use of existing heuristics might become prohibitive.*

1. Introduction

Model composition [Farias 2010] [Dingel 2008] [France 2007] [Clarke 2001] [OMG 2008] [Clarke 2001] plays a crucial role in many software development activities, such as the reuse and evolution of design models. Model composition can be defined as a set of activities performed to combine two input models, M_A and M_B, to produce an output composed model, M_{CM}. The latter often needs to be reviewed and changed to become complaint to an output intended model, M_{AB}. Software developers usually rely on the use of heuristic composition algorithms [Clarke 2001], which match input model elements by automatically "guessing" their semantics. Consequently, the composed and intended models often do not match ($M_{CM} \neq M_{AB}$) in practice because the input models conflict in some way, leading one or more syntactic and semantic inconsistencies [Oliveira 2008]. It is very difficult, if not impossible, to resolve such inconsistencies automatically because this task often requires the understanding of what the model elements mean. Thus, developers might need to invest some effort to resolve emerging inconsistencies; otherwise, the produced design model will represent anything other than what is expected.

Unfortunately, nothing is known about the suitability of state-of-practice composition heuristics on the evolution of design models. Most of the research on model composition is focused on building new model composition strategies (e.g., [Clarke 2001] [OMG 2008]). There is no guidance to help developers to minimize model composition effort, apart from policies for naming model elements and meta-model construction [Meyer 1988] [France 2007]. A possible strategy for avoiding

composition inconsistencies is to structure the models to-be-composed based on design-for-change principles [Meyer 1988]. The assumption is that if a model is well-structured and decomposed with changes in mind, then it should not succumb in the presence of changes when they evolve using heuristics-based composition algorithms.

This paper presents an exploratory study that investigates the impact of model stability on the composition effort. We analyzed the inconsistency rate as well as the resolution effort required to derive successive releases of realistic product-line architecture. Three well-established composition algorithms [Clarke 2001], namely *override*, *merge* and *union*, were employed to evolve product-line architecture along six releases. The initial results, supported by statistical tests, show that it is likely that the more well-designed and stable design is, the lower the inconsistency density and the resolution effort. On the other hand, design for change is not always possible and, in such circumstances, the use of the composition heuristics became either costly or prohibitive; in these contexts, the use of emerging non-heuristic techniques (e.g. [Farias 2010] [France 2007]) might be inevitable.

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

2. Background

2.1. Model Composition Effort

M_A is the current design model while M_B is the model expressing the evolution (delta model), for example, the upcoming changes being added. M_B is inserted into the M_A using composition algorithms, which are responsible for defining the composition semantics and specify how M_A and M_B should be manipulated to produce M_{AB}. We will use the terms composed model (M_{CM}) and intended model (M_{AB}) to differentiate between the output model produced by a composition algorithm and one is desired by the developers. Usually, $M_{CM} \neq M_{AB}$ because the input models conflict in some way. The higher the number of inconsistencies in M_{CM}, the more distant it is from the intended model, M_{AB}. Once M_{CM} has been produced, the next step is to measure the effort to transform M_{CM} into M_{AB} i.e., the effort to resolve inconsistencies. If M_{CM} is equal to M_{AB}, then inconsistency resolution effort is equal to zero. Otherwise, the effort is higher than zero.

2.2. Stability Analysis

M_{CM} can be considered stable if its design characteristics have a low variation with regards to the characteristics of M_{AB}. In our study, we define low variation as equal to or less than 20 percent. This choice is based on previous empirical studies [Kelly 2006] on software stability that has demonstrated the usefulness of this threshold. For example, if the mean of measures of M_{CM} is equal to 9, and the mean of M_{AB} is to 11. So M_{CM} is stable in relation to M_{AB} (because 9 is 18% lower than 11). Following this stability threshold, we can systematically identify whether the M_{CM} keeps stable given

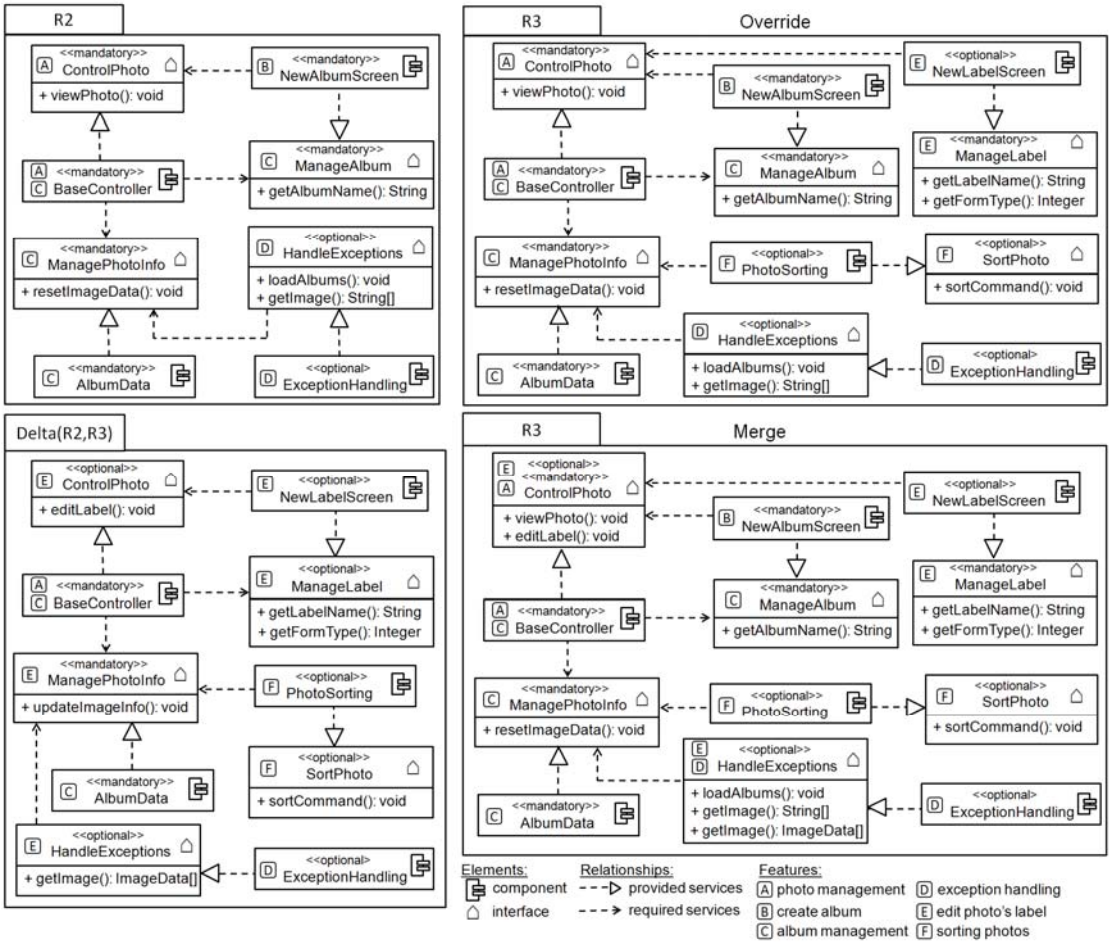

Figure 1 The base and delta model in the third Mobile Media evolution scenario (left). The use of merge and override algorithms (right).

an evolution scenario. However, this threshold was used more as a reference value rather than a final design maker. This difference is computed from the comparison of measures of the model characteristics calculated with a suite of metrics (Section 3.4) [Results 2011].

2.3. Composition Heuristics and Inconsistencies

Composition algorithms rely on two key activities: matching and combining the input design model elements. Note that algorithms are used to modify, remove and add model elements to existing design models. This paper focuses on three well-established composition algorithms: *override*, *merge* and *union* [Clarke 2001]. These algorithms were chosen because they have been applied to a wide range of model composition scenarios such as model evolution [Dingel 2007] [OMG 2008], ontology merge, and conceptual model composition. In addition, they are supported by tools such as IBM Rational Software Architecture [Norris 2011]. Figure 1 shows the application of the composition algorithms in realistic evolution scenario in our study. Two broad categories of inconsistencies [Oliveira 2008] were identified in our study: (1) syntactic inconsistencies, which arise when the composition heuristic results in a model not conforming to the modeling language's metamodel; and (2) semantic inconsistencies,

where the meaning of the composed model does not match that of the intended model. We compute only certain categories of inconsistencies, which are doable to spot manually. A typical example of semantic inconsistency considered in our investigation was the lack of an expected functionality in a model element.

3. Study Methodology

3.1. Objective and Research Questions

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

analyze the stability of design models **for the purpose of** investigating its effects **with respect to** inconsistency rate and developers' effort **from the perspective of** the developers **in the context of** evolution of a software product-line

In particular, this study is aimed at investigating the stability effects on the inconsistency rate and the developers' effort. Thus, we focus on the following two research questions: What is the effect of stability on the inconsistency rate (RQ1) and developers' effort (RQ2)?

3.2. Hypothesis Formulation

3.2.1. First Hypotheses: Effect of Stability on Inconsistency Rate

The first hypothesis is intended to evaluate whether the inconsistency rate in stable design models is significantly different than unstable design models. The intuition is that as design models are well-structured [Meyer 1988] [Martin 2002] and *design-for-change principles* [Meyer 1988] [Martin 2002] were applied, the system decomposition is more modular and resilient to changes (i.e., more stable). Then, it is expected that a more effective modularization can help the composition heuristics (Section 2.3) to better accommodate the upcoming evolving changes into an existing design model; thus minimizing the inconsistency density. These hypotheses are summarized as follows:

Null Hypothesis 1, $H_{1\text{-}0}$:
Stable models have similar or higher inconsistency rate than unstable ones.
$H_{1\text{-}0}$: Rate(stable models) $\geq$ Rate(unstable models).
Alternative Hypothesis 1, $H_{1\text{-}1}$:
Stable models have a lower inconsistency rate than unstable ones.
$H_{1\text{-}1}$: Rate(stable models) **<** Rate(unstable models)

3.2.2. Second Hypotheses: Effect of Stability on Developer Effort

Inconsistencies have a tendency to propagate in a composed model. That is, the introduction of one inconsistency can often lead to multiple other inconsistencies as a result of a "knock-on" effect. An example would be the inconsistency whereby a composed component is missing an important operation. This semantic conflict leads to a "knock-on" syntactic inconsistency if another component requires the operation. In the worst case, there may be long chains of inconsistencies all derived from a single conflict. Studying such propagation effects is important because propagation directly affects the effort in resolving inconsistencies – e.g., a propagation chain of length *n* may actually be fixed by resolving a single conflict rather than the expected *n* conflicts. Thus, we conjecture that design models that are well-structured upfront can isolate these inconsistencies. However, it is by no means obvious that this hypothesis holds. It may

be, for instance, that added changes to the base design model can give rise to unexpected interdependence among design model elements in the output composed model. Consequently, this additional interdependence may significantly increase the developers' effort because additional effort must be invested to restructure the model elements so that the output intended model can be obtained. Thus, we are interested in understanding the possible difference of effort to resolve inconsistencies in stable and unstable design models. The expectation is that stable models may alleviate the developers' effort to produce output intended model using state-of-practice composition heuristics. This leads to the second null and alternative hypotheses as follows:

Null Hypothesis 2, $H_{2\text{-}0}$:
Stable models require similar or higher effort to solve inconsistencies than unstable models.
$H_{2\text{-}0}$: Effort(stable models) ≥ Effort(unstable models).
Alternative Hypothesis 2, H2-1:
Stable models require a lower inconsistency resolution effort than unstable ones.
$H_{2\text{-}1}$: Effort(stable models) < Effort(unstable models).

3.3. Target Case: Evolving a Product Line Architecture

MobileMedia: the Target Software Product-Line. A product line, called MobileMedia [Figueiredo 2008], was selected to be the target case of the evaluation. Model compositions were defined to generate the new releases of the MobileMedia SPL. The purpose of MobileMedia is to provide support for the manipulation of photos, music, and videos on mobile devices. The reasons for selecting this system in the evaluation are: (i) different types of change were realized in each release, including refinements of the architecture style employed; and (ii) the system has been successfully used in other studies involving empirical evaluation of stability in code level. As such, all these factors provided a solid foundation for our study.

3.4. Measured Variables and Quantification Method

Dependent Variables. The dependent variable of hypothesis 1 is the inconsistency rate. It quantifies the amount of composition inconsistencies divided by the total number of elements in the composed model. It is defined from multiple inconsistency metrics, which can be found in [Results 2011]. The dependent variable of the hypothesis 2 is the inconsistency resolution effort—that is, the number of operations (creations, removals, and modifications) needed to transform the composed model into the intended model.

Independent Variable. The independent variable of the hypotheses 1 and 2 is the stability. The measure of stability can be defined as: $S = \{x \in \mathbb{R}_+ / 0 \leq x \leq 1\}$. This measure is mapped to a nominal scale with two categories: Stable Model (SM), if $0.8 \leq S \leq 1$; and Unstable Model (UM), if $0 \leq S < 0.8$. Making use of the design metrics [Results 2011], stability is quantified by the sum of the model characteristics have a variation ≥ 0.8 divided by the total number of characteristics considered (in this case nine). We are interested in the stability with regards to the intended model (M_{AB}). In the second stability, we assess how well the composition algorithms accommodate the changes into the composed model (w.r.t. the intended composition).

3.5. Evaluation Procedures

Composition and Measurement Stages. From one release to another, 6 compositions were produced: 3 compositions following override, merge, and union from the current release to delta model, and 3 compositions in the inverse direction. In total, 60 compositions were performed. The result of this phase was a document of composition descriptions, including the gathered data from the application of our metrics suite. We used a well-validated suite of inconsistency metrics defined in our previous work [Oliveira 2008].

Effort Assessment Stage. The goal of this phase was to assess the effort to resolve the inconsistencies using the quantification method described in Section 3.4. The composition algorithms were used to generate the evolved models, so that we could assess the impact of stability on the model composition effort. In order to support a detailed data analysis, the assessment phase was further decomposed in two main stages. The first stage is concerned with pinpointing the inconsistency rates produced by the compositions (H1). The second stage aims at assessing the effort to resolve a set of previously identified inconsistencies (H2). All measurement results and the raw data are available at [Results 2011].

4. Composition Effort Analysis

4.1. H1: Stability and Inconsistency Rate

4.1.1. Descriptive Statistics

This section describes interesting aspects of the collected data. For this, descriptive statistics for the inconsistency rate are depicted in Figure 2-I. The main conclusion is that stable design models have a lower inconsistency rate. The measures show evidence to support this observation. For instance, the results indicate that stable design models produce an inconsistency rate that, on average, is 63.4 percent lower than the inconsistency rate produced by unstable design models (e.g., a mean of 0.84 compared to a mean of 2.27 for stability related to the intended model). Following the procedures described in Section 3, 60 compositions were performed in total. However, only 15 produced the intended model — that is, the inconsistency rate was equal to zero. Therefore, our results suggest that stable design models have a positive impact on inconsistency rate. Even though, the collected measures had extreme values, they are not considered as true outliers. Hence, they were not removed because they do not tamper the results.

Stable design models present a lower inconsistency rate than unstable design models. This finding is particularly understandable if we take into account previous studies that report positive correlation between low variation of coupling and complexity with design stability [Kelly 2006]. The insights that we can draw out from the descriptive statistics is that: whether models, which are notoriously unstable, have high inconsistency rate because they did not have good design properties, such as high coupling, or because the composition algorithms do not worked well in all evolution scenarios? Observing some findings from previous studies, both insights are true. Kelly [Kelly 2006] states that well-planned software design leads to accommodate the changes in a better way. On the other hand, we observed that well-known composition algorithms (override, merge and union) used in this study are not effective to a set of

(I) DESCREPTIVE STATISTICS

Variables	Groups	N	Mean	Median	St. Dev.
Inconsistency Rate	SM	32	0,84	0,4	1,17
	UM	28	2,27	2	1,25
Effort Rate	SM	32	5.90	4.5	6.75
	UM	28	26.07	27	9.1

(II) MANN-WHITNEY TEST

Mean Rank	Rank Sum
21.04	505
36.81	1325
16.81	403.50
39.63	1426.50

(III) SPEARMAN'S CORRELATION

N	*SC*	*t-value**	*p*
60	-0.542	- 4.92	< 0.001
60	- 0.834	- 11.51	< 0.001

N: number of composed models, St. Dev. Standard Deviation, * with 58 degree of freedom

Figure 2 Descriptive statistics, hypotheses test and correlation analysis

evolution categories such as (i) modification — that is, a model element has some properties modified; and (ii) derivation — that is, model elements are refined and/or moved to accommodate the changes.

4.1.2. Hypothesis Testing

This test evaluates whether in fact the difference between the inconsistency rates of SM and UM groups are statistically significant ($p \leq 0.05$ to indicate a true significance).

Mann-Whitney test. As the collected data violated the assumption of normality, the non-parametric Mann-Whitney test was used as the main statistical test. The results produced are U' = 659.00, U = 205.00, z = 3.418 and $p < 0.001$. The p-value is lower than z so that we can reject the null hypothesis of no difference between the rates of inconsistency of the SM and UM groups ($H_{1\text{-}0}$) — that is, there is sufficient evidence to say that there is a difference between the inconsistency rate measures of the two groups in the Mobile Media project. Figure 2-II depicts that the mean rank of inconsistency rate for UM are higher than that of SM. As Mann-Whitney test relies on ranking scores from lowest to highest, the group with the lowest mean rank is the one that contains the largest amount of lower inconsistency rate. Likewise, the group with the highest mean rank is the group that contains the largest amount of higher inconsistency rate. Hence, the collected data show that unstable models tend to have higher inconsistency rate than the stable design models.

Correlation Analysis. Spearman's correlation (SC) test was played rather than Pearson's correlation because the data set is not normally distributed. It is important to point out that this test assumes that both variables are independent. The correlation coefficient takes on values between -1 and 1. Values close to 1 or -1 indicate a strong relationship between the stability and inconsistency rate. A value close to 0 indicates a weak or non-existent relationship. As can be seen in Figure 2-III, the t-test of significance of the relationship has a low p-value, 0.001, indicating that the correlation is significantly different from zero. Spearman's correlation analysis resulted in a negative and significant correlation (SC = - 0.542). The negative value indicates an inverse relationship — that is, as one variable increases, the other decreases. Hence, composition inconsistencies are more frequently manifested in unstable models rather than stable models. The above correlation suggests that whereas the stability of design models decreases the inconsistency rate in such models increases. Therefore, on average, stable models have significantly lower inconsistency rate than those that are unstable. Thus, we can reject the null hypothesis ($H_{1\text{-}0}$), and confirm the alternative hypothesis ($H_{1\text{-}1}$).

4.2. H2: Stability and Inconsistency Resolution Effort

4.2.1. Descriptive Statistics

Figure 2-I provides the descriptive statistics of sampled inconsistency resolution effort in stable and unstable model groups. There the number of models (N), their mean, median and the standard deviations of each group are presented. From 60 compositions, 53.33 percent of them (N = 32) produced stable design models related to the intended model and the other 46.66 percent (N = 28) produced unstable design models. If we compare the median values of the inconsistency resolution effort of both SM and UM groups, we can observe that SM's median (4.50) is much higher than UM's median (27). Note that this also happens with the mean and standard deviation, which represent the measure of central tendency and measure of dispersion, respectively. Thus, stable design models require less effort than unstable design models by about 77.36 percent (e.g., a mean of 5.9 compared to a mean of 26.07). Moreover, the median of effort is much lower for the stable design models than for unstable design models (e.g., median equal to 4.5 instead of 27). This substantial difference suggests that stable design models require less effort than unstable design models to reach the intended model—that is, a lower amount of operations (creations, removals, and modifications) should be performed to transform the composed model into an intended model.

3.2.2 Hypothesis Testing

This test checks (Figure 2-II) statistically whether the difference between the inconsistency resolution effort required by SM and UM model is significant ($p \leq 0.05$ to indicate a true significance).

Mann-Whitney test. The collected data do not respect the assumption of normality; therefore, the non-parametric Mann-Whitney test was used as the main statistical test as well as it was done in the first hypothesis. The results of the Mann-Whitney test produced are U' = 760.50, U = 103.50, z = 4.949 and $p < 0.001$. The p-value is lower than z, so the null hypothesis ($H_{2\text{-}0}$) can be rejected—that is, there is strong evidence about the difference between the median measures of the two groups. As we can see in Figure 2-II, mean ranks of the measured variables are not similar. We can see that the difference between them in the SM and UM groups is quite significant—that is, the mean rank in the SM group is 57.58 percent lower than the mean rank in the UM group. As Mann-Whitney test relies on ranking scores from lowest to highest, the group with the lowest mean rank is the one that requires the largest amount of lower effort. Likewise, the group with the highest mean rank is the group that contains the largest amount of higher effort. Hence, the collected data show that models that are not stable tend to have higher effort than the stable design models.

Correlation Analysis. As the gathered data do not follow a normal distribution we cannot apply the Pearson's correlation analysis, the Spearman's correlation (SC) test was applied. Figure 2-III provides the results the Spearman's correlation test. The low p-value < 0.001 indicates that the correlation is significantly different from zero. Note that Spearman's correlation value close to 1 or -1 indicates a strong relationship between the stability and effort. On the other hand, a value close to 0 indicates a weak or non-existent relationship. The results (SC = - 0.8341) suggest that there is a negative and significant correlation between the two variables. This implies that whereas the stability

increases the effort to resolve inconsistency decrease. Hence, stable design models tend to require less effort to be transformed into the intended model than unstable design models. Having such results, we can reject the null hypothesis ($H_{2\text{-}0}$), and accept the alternative hypothesis ($H_{2\text{-}1}$): stable design models tend to require lower effort to resolve composition inconsistency than unstable design models. In fact, we have also recently observed this phenomenon in a real-world project (in a different application domain) – based on the use of IBM Rational Software Architecture [Norris 2011]. Despite there is a significant correlation between stability and inconsistency resolution effort, it is difficult (if not impossible) to precisely estimate the effort required when transforming a composed model into an intended model given the problem at hand.

5. Related Work

To the best of our knowledge, our results are the first to empirically investigate the relation between stability and model composition effort. The current model composition literature does highlight the importance of performing empirical studies in model composition [France 2007]. However, nothing has been done up-to-now. For example, the UML built-in composition mechanism, namely package merge [OMG 2008], does not define metrics or criteria to assess the UML models that are merged. Moreover, it has been found to be incomplete, ambiguous and inconsistent [Dingel 2008]. Finally, some previous works investigate the effect of using UML diagram and its profiles with different purpose. In [Ricca 2010], Ricca 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 some typical software engineering activities, none has investigated the peculiarities of UML models in the context of model composition.

6. Threats to Validity

As our study was of exploratory nature, it is not aim here to generalize results. However, we discuss a number of threats that can well inform researchers that plan to replicate our study in more controlled settings. External validity threats concern mainly limitations to generalize the results of the study to a broader context. The main threats are: (i) the use of single target application, and (ii) the use of specific metrics to compute the model composition effort. Obviously, more investigations involving other applications with compositions of larger UML models are required. Moreover, the results might be specific to component models of SPL architectures similar to the one used in this study. Finally, we have minimized the threats by establishing guidelines, periodic reviews with developers of the architectural models, and by engaging them in the discussions.

7. Conclusions and Future Work

In this paper we empirically investigate the impact of stability on model composition effort. The main finding was that the presence of stable model tends to minimize the inconsistency rate and alleviate the inconsistency resolution effort. This observation was derived from statistic analysis of empirical data that have shown a significant correlation between the independent variable (stability) and the dependent variables (conflict rate and effort). The developers can have estimation about both the composition conflicts and their resolution effort given a stability measure of the output composed model.

This paper is the first study that investigates the extent of the impact of design stability on model composition. Hence, further empirical studies are still required to evaluate the impact of stability on model composition in real-world settings. The main reason is that we need to better understand if stable composed models have some gain or not: (i) when produced by other composition algorithms, and (ii) with respect to the time spent to identify the conflicts rather than the effort to resolving them. We hope that the issues outlined throughout the paper encourage researchers to replicate our study in the future under different circumstances.

References

Clarke, S., Composition of Object-Oriented Software Design Models, Ph.D. Thesis, Dublin City University, January, 2001.

Dingel, J., Diskin, Z., and Zito, A., Understanding and Improving UML Package Merge. Journal of SoSym, 7(4):443–467, 2008.

Farias et al. Empirical Evaluation of Effort on Composing Design Models. In: ICSE's Doctoral Symposium, pp. 405-408, 2010.

Figueiredo et al. Evolving Software Product Lines with Aspects: An Empirical Study on Design Stability. In: ICSE'08, pp. 261–270, 2008.

France, R. and Rumpe, B., Model-Driven Development of Complex Software: A Research Roadmap. In: FuSE at ICSE'07, pp. 37–54, 2007.

Kelly, D., A Study of Design Characteristics in Evolving Software Using Stability as a Criterion, IEEE TSE, 32(5):315–329, 2006.

Martin, R., Agile Software Development, Principles, Patterns, and Practices, Prentice Hall, 2002.

Meyer, B., Object-Oriented Software Construction, 1st ed. Prentice-Meyer, Hall, Englewood Cliffs, 1988.

Norris, N., and Letkeman, K., Governing and managing enterprise models: Part 1. Introduction and concepts, IBM Developer Works, http:// www.ibm.com/developerworks/rational/library/09/0113_letkeman-norris, 2010.

Oliveira et al. On the Quantitative Assessment of Class Model Compositions: An Exploratory Study. In: ESMDE at MODELS, 2008.

OMG. Unified Modeling Language: Infrastructure version 2.2. Object Management Group, February 2008.

Results, Evaluating the Effects of Stability on Model Composition Effort: an Exploratory Study, http://www.kleinnerfarias.com/eselaw2011, 2011.

Ricca et. al., How Developers' Experience and Ability Influence Web Application Comprehension Tasks Supported by UML Stereotypes: A Series of Four Experiments, IEEE TSE, 96(1):96–118, 2010.

Wohlin et. al., Experimentation in Software Engineering: an Introduction, Kluwer Academic Publishers, Norwell, USA, 2000.