On the Quantitative Assessment of Class Model Compositions: An Exploratory Study

Kleinner Oliveira[1], Alessandro Garcia[2], Jon Whittle[2]

[1] Computer Science Department
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, RJ - Brazil
kleinner@gmail.com

[2] Computing Department
Lancaster University – InfoLab 21
Lancaster - UK
{alessandro,jon}@comp.lancs.ac.uk

Abstract. Model composition can be viewed in model-driven engineering as an operation where a set of activities should be performed to merge two input models into a single output model. The latter aggregates syntactical and semantic properties from the original models. However, given the growing heterogeneity of model composition strategies, it is particularly challenging for designers to objectively assess them given a particular problem at hand. The key problem is that there is a lack of canonical set of indicators to quantify harmful properties associated with the output models, such as composition conflicts and modularity anomalies. This paper presents an inquisitive study in order to capture an initial set of metrics for assessing and comparing model composition strategies in two case studies. We apply a number of metrics to quantify different conflict types and modularity properties arising at composite class models produced with override and merge-based strategies. We have observed that some of the quantitative indicators were effective to pinpoint when a model composition strategy is not properly chosen. In some cases, the output models exhibited non-obvious undesirable conflicts and anti-modularity factors.

Keywords: Model Composition, MDE, Metrics, Assessment.

1 Introduction

Given the central role that model composition plays in model-driven engineering nowadays, researchers are increasingly focusing on defining and improving alternative techniques for composing structural or behavioural models. Model composition can be defined by a composition operation, a special type of model transformation, that takes two models Ma and Mb as input models and combines their elements into an output model Mab. Several mechanisms have been proposed in order to put model composition into practice (e.g., see [2, 3, 4, 5, 6]), based on related work

in many different domains, such as database integration [7], aspect-oriented modeling, model transformation , and merging of state charts.

However, not much attention has been paid to the quality assessment of such model composition techniques. Even worse, according to [5] there is very little experience that can be used to determine the worth of current approaches. Given the growing heterogeneity of model composition strategies [3], such as *override* and *merge*, it is intrinsically difficult to systematically quantify undesirable phenomena that arise in the output composite models, including abstract syntax conflicts and semantic clashes. It is particularly challenging for researchers or designers to objectively assess the output model and the composition strategy itself given the problem at hand.

In this paper we start to tackle such needs through an exploratory study (Section 2) on assessing composition strategies for class models. The goal is to inquisitively identify an initial set of indicators for the evaluation and comparison of alternative composition strategies. We have applied a metrics suite (Section 3) to quantify the conflicts rate and modularity properties arising in class model compositions based on merge and override. Our long-term goal is to define a comprehensive assessment framework intended to guide researchers and designers on the assessment of model composition techniques. In our study, we have detected that some of the used quantitative indicators were effective to determine when a model composition strategy is not properly chosen (Section 4). In certain cases, the output models exhibited non-obvious syntactic and semantic conflicts and a number of modularity anomalies not existing in the original input models. We also contrast the initial findings of our exploratory investigation with related work (Section 5). Finally, we present some concluding remarks (Section 6).

2 Experimental Procedures

This section describes the experimental procedures used in our exploratory study. Two case studies were performed in order to investigate possible problems associated with the use of composition strategies for class models. The first study comprises a set of real-life models for an Automated Highway Toll System. In this case, different members of a distributed software development team were in charge of modeling different use cases of the system. They would need to cope with model composition problems when bringing the use cases together.

There are three packages, namely (for simplification) Packages A, B, and C, where each of them implements a set of use cases. There are two explicit compositions defined for these packages (Figure 1). Package A presents a UML class diagram that specifies basically functionalities related to: create user account, add funds, and stop toll booth. Package B specifies functionalities related to: synchronizes accounts, process credit card, transponder and vehicle. While the Package C specifies functionalities related to add transponder and start toll booth. The goal is to produce an output Package that gathers all functionalities together. To this end, we need to merge the Package A, B and C according to a particular composition strategy (*override* or *merge* specifically). The choice of a particular composition strategy is

very important to produce sound output models while not introducing modularity impairments.

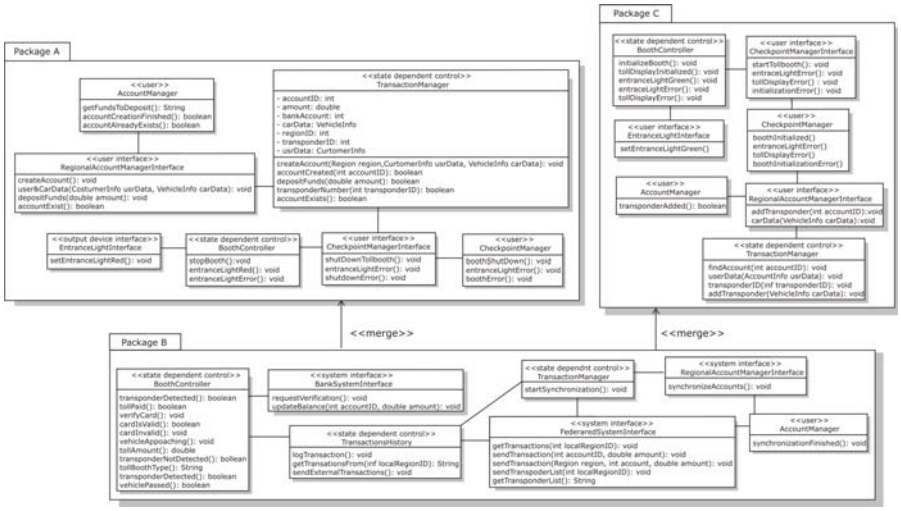

Figure 1. Example of composition of an automated highway toll system.

The second study consists of a literature-based [11] example of a calculator that is depicted in Figure 2. It has two packages: (1) Package A presents a UML class diagram that specifies a Calculator to implement two basic functionalities: sum and subtraction; and (2) Package B represents a Calculator that implements three functionalities: sum, division, and multiplication. The goal is to produce an output Calculator that contains four operations: sum, subtraction, division, multiplication. To do this, we need to put these functionalities together in a single Package by merging Package A and Package B.

Our aim is to assess in which ways the composition strategies (*override* and *merge* specifically) impact on the input models' properties. The *merge strategy* usage is more appropriate when the input design models contain specifications for different requirements of a software system. On the other hand, the *override strategy* can be indicated when elements in an existing model need to be somehow evolved or changed. The semantics of the override strategy [3] can be briefly defined as: (i) for all elements in the Package A and Package B that are corresponding, the Package A's element should override its corresponding element; and (ii) elements in the Package A and B that are not involved in a correspondence match remain unchanged and they are inserted into the output model (Package AB).

The semantics of the merge strategy [3] can also be defined as: (i) for all elements in the Package A and Package B that are corresponding elements, they should be combined; and (ii) elements in the Package A and B that are not involved in a correspondence match remain unchanged and they are inserted into the output model (Package AB). However, when we put these elements together in the output model (as the result of either overriding corresponding elements or adding elements in

the Package AB directly) may result in some problems such as semantic clashes. We will propose a metrics suite to provide ways to assess how useful or harmful such composition relationships are following a specific composition strategy. The goal is to provide initial support for designers and researchers objectively analyze which composition strategy minimizes the conflicts rate while maximizing modularity benefits in the output model.

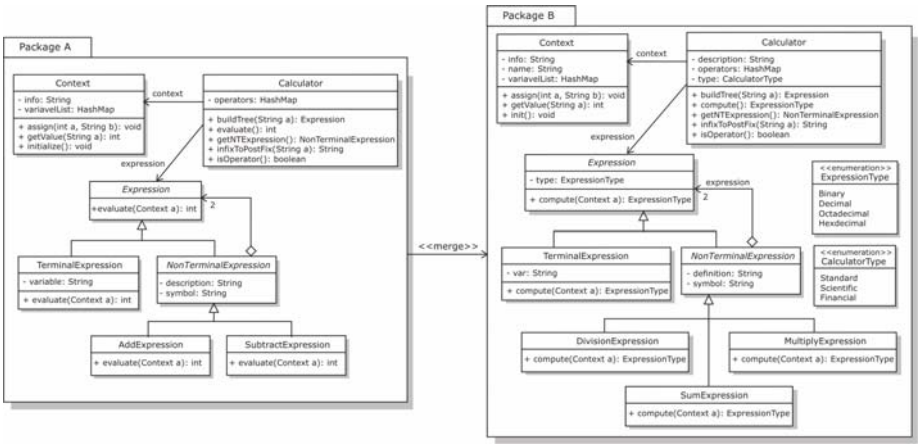

Figure 2. Example of composition of calculators

3 A Metrics Suite for Model Composition

This section presents the metrics suite defined for assessing the model compositions in our exploratory study. This framework guides the researchers for assessing and coping with difficulties of UML model composition assessment.

3.1 Quantifying Composition Conflicts Rate

Number of Abstract Syntax Conflicts **(NAbSC)**
This metric counts the number of *abstract syntax conflicts* in a class model. *Abstract syntax conflicts* occur when a model does not comply with the UML metamodel' metaclasses and their structural relationships. It is a well-known problem, for instance, in graph transformations. The goal is to quantify and check inconsistencies of the target models against the UML metamodel. Once all the conflicts have been addressed (i.e. NAbSC = 0), the output model can be considered as compliant to he UML metamodel. Otherwise, the output model is an invalid or non-compliant model.

$$NAbSC = \sum_{i=1}^{SM} k_i \ ,$$

where:
SM – a set of model elements.
k_i – the number of AbSC of the i-th model element.

***Number of Semantic Clash Conflicts (*NSCC)**
This metric counts the *number of semantic clash conflicts* in a model. A semantic clash conflict occurs when model elements have different names, however, with same semantic value. We need to quantify such conflicts in order to identify unexpected semantic clash problems in the output models. For instance, models with semantic clashes may become ambiguous and inconsistent. In addition, it may affect the model understandability or complicate some tasks such as model transformation and code generation. If the NSCC has a high value, it may imply that the output model is useless. This metric is given by the formula:

$$NSCC = \frac{1}{2}\sum_{i=1}^{SM} w_i \ ,$$

where:
SM – a set of model elements.
w_i – a boolean value that represents if an i-th model

***Number of Compositions of a Model Element* (NCME)**
This metric counts the number of compositions that a model element has participated. The number of compositions may be an effective indicator of *semantic mix conflict*. When model elements are composed, their semantics are mixed and it may lead to unsound model elements. For example, a design pattern assigns roles to their participant classes, which define the functionality of the participants in the pattern context. When UML class diagrams are merged such roles may be modified having negative impacts on quality attributes of the design pattern. This metric is given by the formula:

$$NCME = |M| \ ,$$

where:
M – the number of compositions that a model element has participated during the composition process.

***Number of Behavioral Feature Conflicts* (NBFC)**
This metric counts the number of behavioral feature conflicts in a class. A *behavioral feature conflict* may occur when a class: (1) has two (or more) methods that are used with the same purpose, and (2) refers to a method that no longer exists, or exists under a different behavior that is not expected. The high NBFC measure may represent some undesirable model composition phenomena. This metric is determined by the formula:

$$NBFC = |B| \ ,$$

where:
B – the number of behavioral feature (method) conflicts in a class

***Number of Unmeaning Model Elements* (NUME)**
This metric counts the number of unmeaning model elements in a model. During the composition process, the model elements are manipulated and sometimes some elements are not referred nor make reference to other elements, that is, they are isolated. This metric is given by the formula:

$$NUME = |U| \ ,$$

where:
U – the number of unmeaning model element in a mode.

3.2 Quantifying Modularity Anomalies in Composite Models

We have also applied some classical metrics intended to measure some modularity-related characteristics of a class, such as coupling degree, number of attributes, and operations. These metrics are described in Table II. Due to space constraints, these metrics are briefly presented. In fact, most of these metrics (e.g. NATC and CBC) were originally defined by other authors and their definitions can be found in their respective publications [14, 17]. The goal of using these metrics is to assess how the composition process affects the output models regarding some design principles, such as low coupling, when we specify different composition strategies. In addition, in many cases, composition strategies can artificially lead to the introduction of design anomalies ("bad smells"), such as "Temporary Field"; this bad smell can be identified comparing the NATC of a class in the output model against the respective classes in the input models used for the composition.

Table I. The Class-level Modularity Metrics

Metric	Description
Number of Attributes in a Class (NATC)	Counts the number of attributes in a class.
Number of Operations in a Class (NOPC)	Counts the number of operation in a class.
Number of Associations between Classes (NASC)	Counts the number of associations per class; the new language produced from a model composition may not be consistent with the domain defined previously.
Coupling between Classes (CBC)	Counts the number of all dependencies of a class to other classes in the system.
Number of Subclasses of a Class (NSUBC)	Counts the number of children of a class.
Number of Superclasses of a Class (NSUPC)	Counts the parents of a class.

4. Results and Discussion

Quantitative assessment is an effective way to supply measures and evidence that may improve our understanding about model-driven engineering techniques, in our case, model composition. Although quantitative studies have some disadvantages, they are very useful because they boil a complex situation down to simple numbers that are easier to grasp and discuss. This section provides a general analysis and discussion of the data that have been collected from applying the set of defined metrics to model compositions derived in the two case studies (Section 2).

Graphics are used to represent the data gathered in the measurement process. The Y-axis presents the absolute values gathered by the metrics. Each pair of bars is attached to an integer value, which represents the measure. The X-axis specifies the metric itself. These graphics help analyzing how the composition of the input models affects (or not) the output model regarding a particular metric. These graphics support an analysis of how the change of composition strategy affect (or not) the output model. The results shown in the graphics were gathered according to the model point

of view; that is, they represent the total of metric values associated with all the model elements for each model (output model) that is being considered.

Figure 3 depicts the overall composition results between Package A, B and C of the Automated Highway Toll System following the override and merge strategy. We compare the output model produced by the override and merge strategy and it is possible to observe that no measure was detected to the metrics such as NSUNC, NSUPC, NAbSC, and NSCC. On the other hand, the NOPC metrics have a higher measurement following merge strategy than override strategy. This observation can indicate a negative point considering reusability. Although not showing differences between each other with regard to the NASC, NSUBC, and NSUPC metrics, the output models presents significant differences, for example, the Package BAC produced by the merge strategy presents all functionalities defined in the Package A, B and C, while the Package BAC produced by the override strategy contains only functionalities defined in the Package B.

According to the measures concerning the number of associations between classes (NASC), the number of abstract syntax conflicts (NAbSC), and the number of subclasses (NSUBC) and superclasses of a class (NSUPC), no significant difference was detected in favor of a specific composition strategy when applied to the two case studies. The measures of NSUBC and NSUPC can be easily explained because both case studies do not exhibit a hierarchy-depth in their inheritance relationships. On the other hand, the measure of NASC supplies evidence of that number of associations of a Class contains is independent of type of composition strategy.

Package BAC produced by the override strategy provided higher results in two measurements, NCME and NUME. When *EntranceLightInterface* is inserted in the Package BAC this class becomes unmeaning, because of the class that it makes relationship, *PackageA.BoothController*, no longer exists (*PackageA.BoothController* is overridden by the *PackageB.BoothController*). The NASC and CBC measurement have same values. So the coupling in the Package BAC is independent of the kind of composition strategy in this case.

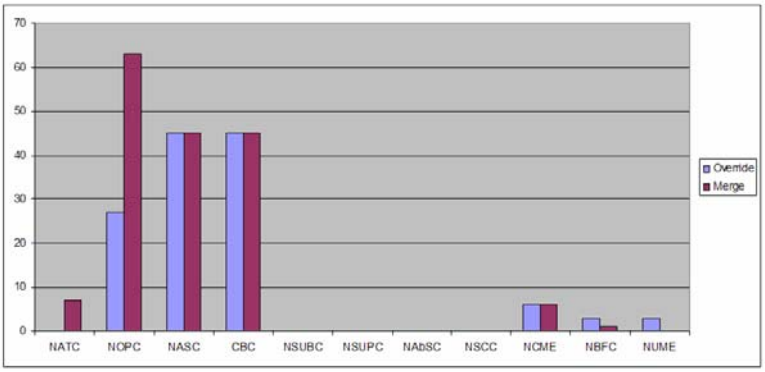

Figure 3. Comparison between output models produced following override and merge strategy.

The measurement regarding the output Calculators has provided some results that are depicted in Figure 4. We compare the output Calculators produced following override and merge strategy. Although not showing differences between each other regarding the NASC, NSUBC, and NSUPC metrics, the output models present significant differences. The Package AB produced by the merge strategy has higher values for some metric measures such as NATC, NOPC, CBC, NSCC, NCME, and NBFC. On the other hand, Package AB produced by the override strategy provided higher results in one only measure, NUME, because two enumerations, *CalculatorType* and *ExpressionType,* are unmeaning in the Package.

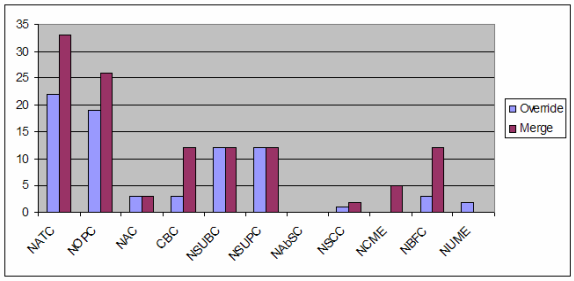

Figure 4. Comparison between calculators produced following override and merge strategy.

According to the data gathered, the most useful metrics in this exploratory study were as follows. First, number *of semantic clash conflicts* (NSCC) as it indicated the presence of a significative number of negative semantic clashes. This measure served as warnings of not helpful output models using a particular composition strategy through the identification of ambiguity and inconsistency arising in semantic clashes. The observation of Figure 3 provides evidence of the effectiveness of this metrics. Second, *number of unmeaning model elements* (NUME) supplied evidence that override strategy is potentially harmful when used beyond the purposed of evolving or changing an existing model. The output models, based on override-driven compositions, had elements that are not referred nor make reference to other elements, that is, they are isolated (unmeaning in the package). Thus, regarding this metric the better strategy to be applied in the case studies was the merge strategy.

Finally, after observing all the conflict rate and modularity results, the metrics indicated that the merge strategy is the best strategy to be used in our two case studies. This finding is also mainly based on the measures of NUME and NSCC (discussed above). Moreover, we should highlight that, as expected, it is particularly challenging for researches to objectively assess the output models and identify conflicts associated with some metrics such as NUME, NAbSC and NSCC. Therefore, the issue of improving automated support for measuring conflict rates should be a topic of future work.

5. Related Work

There is little related work focusing on either the quantitative assessment of models in general or on the quantitative assessment of model compositions. Up to now, most

approaches involving model composition rest on subjective assessment criteria. Even worse, they lead to dependence on experts who have built up an arsenal of mentally-held indicators to evaluate the growing complexity of design models in general [5]. As a consequence, the truth is that modelers ultimately rely on feedback from experts to determine "how good" the input models and their compositions are. 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.

To the best of our knowledge, the need for assessing models during a model composition process neither have been pointed out nor even proposed by current model composition techniques [2, 3, 4, 8, 9]. For example, the UML built-in composition mechanism, namely *package merge*, does not define metrics or criteria to assess the merged UML models. Moreover, it has been found to be incomplete, ambiguous and inconsistent [6].

The lack of quantitative indicators for model compositions hinder our process of understanding better side effects peculiar to certain model composition strategies. Many different types of metrics have been developed during the past few decades for different UML models. These metrics have certainly helped designers analyze their UML models to some extent. However, as researchers' focus has shifted to the activities related to model management (such as model composition, evolution and transformation), hence the shortcomings and limitation of UML model metrics have become more apparent. Some authors [1, 12, 13-18] have proposed a set of metrics that consider UML model's properties. These works have shown that their measures satisfy some properties expected for good measures of design models. However, these metrics can not be employed to assess problems that may arise in a model composition process such as semantic clashes.

6. Concluding Remarks and Future Work

If models are seen as primary development and transformation artifacts in model-driven engineering, then software designers naturally become concerned with how their quality is evaluated. In order to be considered for use in mainstream software development, model composition techniques should be supplemented with quality criteria and indicators. These elements are fundamental for developing and analyzing composition processes and output models. We presented an exploratory study and an initial metrics suite for assessing class model compositions generated by a selected set of model composition strategies. Such metrics are applied to output models and some analysis are performed according to the data gathered.

Our initial evaluation has demonstrated the feasibility of our candidate set of metrics for quantifying modularity properties and conflict rates in composition processes. Obviously, more investigations on its applicability to large UML model compositions are required. Further empirical evaluations are indeed fundamental to validate our quantitative indicators in real-world design settings involving UML model compositions. Thus, future work will concentrate on designing and carrying out a family of empirical studies to assess, for example, compositions of the most popular OMG's UML profiles in realistic scenarios. Finally, we should point out that

model composition assessment is in initial stage and there is very little experience that can be used to determine the feasibility of current approaches. Moreover, its empirical-driven improvement, supported by a comprehensive set of well-validated metrics suite, is absolutely necessary to the evolution of model-driven engineering field. This work represents one of the first stepping stones towards this end.

References

1. J. Aranda, N. Ernst, J. Horkoff, and S. Easterbrook. A Framework for Empirical Evaluation of Model Comprehensibility. In International Workshop on Modeling in Software Engineering (MiSE), pp. 20-26, May, 2007.
2. G. Brunet, M. Chechik, S. Easterbrook, S. Nejati, N. Niu, and M. Sabetzadeh. A Manifesto for Model Merging. In International Workshop on Global Integrated Model Management (GaMMa'06), pages 5–12, Shanghai, China, May 2006. ACM Press.
3 S. Clarke and R. Walker. Composition Patterns: an Approach to Designing Reusable Aspects. 23rd Intl. Conf. on Software Engineering, pp. 5–14, Toronto, Canada, 2001.
4 T. Cotternier, A. van den Berg, and T. Elrad. Modeling Aspect-Oriented Composition. In 7th International Workshop n Aspect-Oriented Modeling co-located with (MODELS' 05), Montego Bay, Jamaica, October 2005.
5 R. France and B. Rumpe. Model-Driven Development of Complex Software: A Research Roadmap. In Future of Software Engineering (FOSE'07) co-located with ICSE'07, pages 37–54, Minnesota, EUA, May 2007.
6 OMG, Unified Modeling Language: Infrastructure version 2.1, Object Management Group, February 2007.
7 E. Rahm and P. Bernstein. A Survey of Approaches to Automatic Schema Matching. VLDB Journal: Very Large Data Bases, 10(4):334–350, 2001.
8 R. Reddy, R. France, S. Ghosh, F. Fleurey, and B. Baudry. Model Composition – a Signature-Based Approach. In Aspect Oriented Modeling (AOM) Workshop, Montego Bay, Jamaica, October 2005.
9 Y. Reddy, R. France, G. Straw, N. M. J. Bieman, E. Song, and G. Georg. Directives for Composing Aspect-Oriented Design Class Models. Transactions of Aspect-Oriented Software Development, 1(1):75–105, 2006.
10 B. Selic. The Pragmatics of Model-Driven Development. IEEE Software, 20(5):19–25, 2003.
11. Gamma, E. et al. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, 1995.
12. Shyam R. Chidamber and Chris F. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493, June 1994
13. A. Baroni, Quantitative assessment of UML dynamic models, SIGSOFT Software Eng. Notes, vol. 30, no. 5, pp. 366-369, 2005.
14. Baroni, A.L., Abreu, F.B. and Guerreiro, P. The State-of-the Art of UML Design Metrics. Technical Report, Universidade Nova de Lisboa, Monte da Caparica, 2005.
15. Shyam R. Chidamber and Chris F. Kemerer. A metrics suite for object oriented design. IEEE Transactions on Software Engineering, 20(6):476–493, June 1994
16. A. Baroni, Quantitative assessment of UML dynamic models, SIGSOFT Software Eng. Notes, vol. 30, no. 5, pp. 366-369, 2005.
17. A. Baroni, F. Abreu, and P. Guerreiro. The State-of-the Art of UML Design Metrics. Technical Report, Universidade Nova de Lisboa, Monte da Caparica, 2005.
18. M. Genero, M. Piattini-Velthuis, J. Lemus, and L. Reynoso Metrics for UML Models, UPGRADE, vol. 5, number 2, April, 2002.