Assessing the Impact of Aspects on Model Composition Effort

Kleinner Farias, Alessandro Garcia
OPUS Research Group, Department of Informatics,
Pontifical Catholic University of Rio de Janeiro,
Rio de Janeiro - RJ - Brazil
{kfarias, afgarcia}@inf.puc-rio.br

Jon Whittle
Computing Department,
InfoLab21, Lancaster University,
Lancaster, LAI 4AW, UK
whittle@comp.lancs.ac.uk

ABSTRACT
Model composition is a common operation used in many software development activities—for example, reconciling models developed in parallel by different development teams, or merging models of new features with existing model artifacts. Unfortunately, both commercial and academic model composition tools suffer from the *composition conflict problem*. That is, models to-be-composed may conflict with each other and these conflicts must be resolved. In practice, detecting and resolving conflicts is a highly-intensive manual activity. In this paper, we investigate whether aspect-orientation reduces conflict resolution effort as improved modularization may better localize conflicts. The main goal of the paper is to conduct an exploratory study to analyze the impact of aspects on conflict resolution. In particular, model compositions are used to express the evolution of architectural models along six releases of a software product line. Well-known composition algorithms, such as *override*, *merge* and *union*, are applied and compared on both AO and non-AO models in terms of their conflict rate and effort to solve the identified conflicts. Our findings identify specific scenarios where aspect-orientation properties, such as obliviousness and quantification, result in a lower (or higher) composition effort.

Categories and Subject Descriptors
D.2.8 [**Software Engineering**]: Metrics-*Product* metrics; D.2.13 [**Software Engineering**]: Reusable Software.

General Terms
Measurement, Design, Experimentation

Keywords
Model Composition, Software Metrics, Empirical Studies, Software Architecture, Software Product Lines.

AOSD'10, March 15–19, 2010, Rennes and St Malo, France

1. INTRODUCTION
Model composition plays a central role in many software engineering activities—e.g., reconciling models developed in parallel by different development teams, and evolving models to add new features. In collaborative software development, for example, separate developer teams may concurrently work on a partial model of the overall architecture to allow developers to concentrate more effectively on parts of the architecture relevant to them. However, at some point, it is necessary to bring these models together to generate a "big picture" view of the overall architecture. For this reason, there has been a significant body of research into defining model composition algorithms in the areas of model versioning control, software product lines [1], and aspect-oriented modeling [26].

In practice, however, model composition is a highly-intensive manual task[1] [33]. Models to-be-composed inevitably conflict with each other and these conflicts must be *detected* and *resolved* in order to produce a correctly composed model. It is very difficult, if not impossible, to resolve conflicts automatically because conflict resolution relies on an understanding of what the models actually mean and such semantic information is typically not included in any formal way in the models. This need has been the key point of improvement required in enterprise modeling tools that support model composition, such as IBM Rational Software Architecture [33].

The hypothesis of this paper is that aspect-orientation may alleviate the effort of conflict resolution to some extent. The intuition is that aspect-orientation brings improved modularity and that a more effective modularization may localize conflicts, thus making them easier to detect and deal with. However, it is by no means obvious that this hypothesis holds. It may be, for instance, that conflicts in aspect-oriented models have a detrimental effect on conflict resolution effort because aspect conflicts may require the modeler to examine all points in the model crosscut by the aspect. The goal of this paper, therefore, is to report on an exploratory empirical study which aimed to provide evidence to support or refute this hypothesis. We only study one facet of model composition in this paper: the use of

[1] Anecdotal evidence from industry contacts suggests that model composition and conflict resolution is a full-time job.

model composition in adding new features to a set of models for an industrial software product line.

Software product lines (SPLs) commonly involve model composition activities and, while we believe the kinds of model composition in SPLs are representative of the broader issues, we make no claims about the generality of our results beyond SPL model composition. We show the results for model compositions of six releases of an SPL. In each release, models for the new feature are composed with the models for existing features. We analyze, for each release, the quantity and nature of the composition conflicts. Furthermore, we compare two versions of the SPL models—one which uses aspect-oriented modeling (AOM) and one which does not.

Our initial results show that higher conflict rates were observed in the presence of aspects when they had higher degree of quantification. On the other hand, this problem did not entail more effort on conflict resolution. We also found that higher degree of obliviousness tended to yield compositions of AO composed models that are closer to the intended compositions. To the best of our knowledge, our results are the first to empirically investigate potential advantages of aspects *during modeling*. Despite a wide variety of technical approaches to AOM (e.g., MATA [27], Kompose [28]), to-date there has been almost no empirical evaluation of AOM. We therefore see this paper as a first step in a more ambitious agenda to empirically assess aspect-oriented modeling.

The remainder of the paper is organized as follows. In Section 2, we introduce the main concepts and knowledge that are going to be used and discussed throughout the paper. In Section 3, we present the methodology. In Section 4, we present the composition analysis effort. In Section 5, we contrast our work with related work and present the threats to validity in Section 6. Finally, in Section 7, we present some concluding remarks and future work.

2. BACKGROUND

Model composition has been studied for many years in software engineering [7] and in other related disciplines. Model composition can be applied to different contexts, such as model reuse, resolution of stakeholders' viewpoints, and model evolution [21] [22]. For instance, in the context of our exploratory study, model composition is exploited to describe the evolution of an SPL's architectural specification. Section 2.1 defines model composition more precisely and defines how to quantify the effort involved in applying model composition. Our paper compares model composition in an AO and non-AO setting. Section 2.2 describes the aspect-oriented modeling notation used in our experiment. Section 2.3 discusses state-of-practice algorithms for model composition, and Section 2.4 explains recurring types of model composition conflict.

2.1 Model Composition

Model Composition. The term *model composition* refers to a set of activities that should be accomplished to combine two (or more) input models, M_A and M_B, in order to produce an output composite model, M_{AB}. A model *composition algorithm* defines the semantics of the model composition relationship and specifies how the input models should be manipulated in order to compose them. Note that composition algorithms are often heuristic because they rely on matching elements in the input models, which, in turn, relies on "guessing" the semantics of model elements. We will use the terms *composed model* (M_{CM}) and *intended model* (M_{AB}) to differentiate between the composition produced by a composition algorithm and the composition that the developer desires. Usually, $M_{CM} \neq M_{AB}$ because the input models conflict in some way. The number of conflicts may depend on the choice of algorithm but will typically be nonzero if the algorithm is heuristic.

Composition Effort. There is currently very limited knowledge regarding the amount of effort required to apply model composition algorithms. Anecdotal evidence from companies suggests that the effort is significant but, in this paper, we aim to quantify the effort more precisely. The composition effort can be calculated using the equation defined in Figure 1. The equation gives an overview of how composition effort can be measured and what part we focus our study on. The equation makes it explicit that the model composition effort includes: (1) the effort to apply a model composition algorithm: $f(M_A,M_B)$; (2) the effort to detect undesirable conflicts in the output model: $diff(M_{CM},M_{AB})$; and (3) the effort to resolve conflicts: $g(M_{CM})$. Once a composed model (M_{CM}) has been produced, the next step is to measure the effort to transform M_{CM} into the intended model (M_{AB}). If M_{CM} is equal to M_{AB}, then $diff(M_{CM},M_{AB}) = 0$ and $g(M_{CM}) = 0$. Otherwise, $diff(M_{CM},M_{AB}) > 0$ and $g(M_{CM}) > 0$. Our study focuses on assessing the effort to resolve conflicts (i.e., $g(M_{CM})$). We defer consideration of $diff(M_{CM},M_{AB})$ to future work.

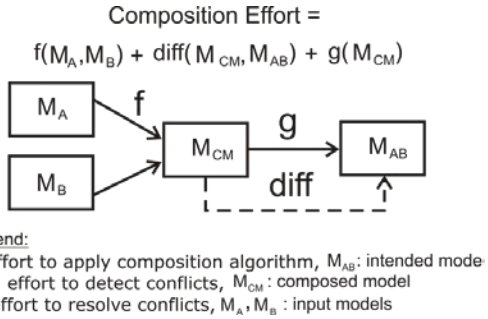

Figure 1. Model composition effort equation.

2.2 Aspect-Oriented Modeling

Model composition applies both to development with and without aspect-oriented modeling (AOM). In fact, in this paper, we compare the model composition effort in both cases. AOM languages aim at improving separation of concerns by supporting the modular representation of concerns that cut across multiple software modules. Crosscutting concerns are represented by a new model element, called *aspect*. The goal of AOM is to provide software developers with the means to express aspects and crosscutting relationships in their models. There are AOM languages for modeling aspects at many levels of abstraction, ranging from use cases and architectural design to detailed designs. As far as the solution space is concerned, aspects are usually first expressed in architectural models.

Figure 2 is an illustrative example of the architectural AOM language [26] used in our study (Section 4). We chose this AOM language because: (i) we selected architectural models as our focus due to the availability of existing industrial models;

(ii) the AOM language has been widely used in other contexts and is therefore mature [26]. The notation (see Figure 2) supports the visual symmetric representation of aspect-oriented software architectures. The target modeling approach consists of an extension of the UML's component diagram [19]. In order to put the composition in practice, we should consider the properties of model elements defined in the UML metamodel specification for this diagram. Thus, the properties of the model elements considered were: component (name, provided interface and required interface), interface (name, operation and attribute), operation (name, return type and parameters), attribute (name and type), relationship (source and target), crosscutting relationship and join-points. Therefore, the composition algorithms (Section 2.3) are fine-grained due to take into account these properties in each composition.

The notation provides explicit elements for expressing different forms of component-aspect collaborations, which are represented by *aspectual connectors*. Aspectual connectors are represented by rectangles in Figure 2 and define which components, interfaces or specific operations are affected by a component modularizing a crosscutting concern. Aspectual connectors are associated with crosscutting relationships represented by dashed arrows. The notation also supports the visual modeling of specific pointcut designators (e.g., advising all the provided interfaces) and sequencing operators (after, before, and around). For the sake of simplicity in this paper, only aspectual connectors and crosscutting relationships will be represented in the models of our case study; all the other visual details have been omitted from here on.

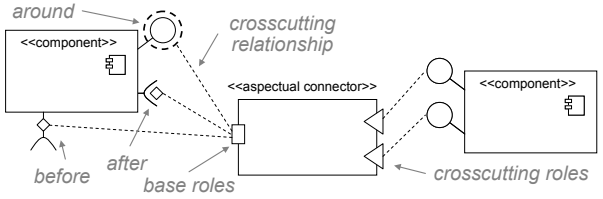

Figure 2. AOM Language for Architectural Models

2.3 Composition Algorithms

Composition algorithms consist of two key activities: matching model elements in the input models, and composing elements. This paper focuses on three well-established composition algorithms: *override*, *merge* and *union*. These algorithms were chosen because they have been applied in a wide range of model composition scenarios, such as model evolution, ontology merge, and conceptual model composition. In addition, they have been recognized as candidate algorithms in aspect-oriented model composition (e.g., Theme/UML [6] [13]).

In the following, we briefly define these three algorithms, where we assume two hypothetical input models, M_A and M_B. We say that two elements from M_A and M_B respectively 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.

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 corresponding elements. Elements not involved in the correspondence remain unchanged and are inserted into the output model.

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 components 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 match remain unchanged and are inserted into the output model directly.

Union. For all elements in the M_A and M_B that are corresponding elements, they should be manipulated in order to preserve their distinguished identification; it means that they should coexist in the output models with different identifiers; elements in the M_A and M_B that are not involved in a correspondence match remain unchanged and they are inserted into the output model, M_{AB}.

2.4 Syntactic and Semantic Conflicts

Conflicts arise in an output model when the composed model (M_{CM}) does not match the intended model (M_{AB}). For instance, a conflict occurs when the functionality of M_{CM} and M_{AB} is different. In practice, we can identify two broad categories of conflicts: (1) syntactic conflicts, which arise when the composition algorithm results in a model not conforming to the modeling language's metamodel[2]; and (2) semantic conflicts, where the meaning of the composed model does not match that of the intended model. It is important to emphasize that, in our study, we count only certain categories of conflicts which are easy to spot manually. For example, it is impossible to automatically count all semantic conflicts. A typical example of semantic conflict considered in our investigation was when functionality expected in a model element was not found. For instance, this is the case when the component does not present the expected functionalities defined in a new release or presents undesirable functionalities.

3. METHODOLOGY

This section describes the running hypotheses for our study (Section 3.1), the target application (Section 3.2), the evaluation method used for computing model composition effort (Section 3.3), and the other study procedures (Section 3.4) in our exploratory study.

3.1 Hypotheses

Aspect-oriented modeling has been a topic of research for at least ten years [13]. However, there is currently very limited knowledge as to how aspects, when incorporated in input models, affect the model composition effort. In particular, there is no understanding as to what extent the composition (Section 2.3) of aspect-oriented models (Section 2.2) affects the emergence of conflicts in composed models.

In this context, the hypotheses of this study can be derived from two key research questions:

[2] One might initially think that there should be no syntactic conflicts. However, many composition algorithms do in fact result in syntactically incorrect models. This is a well-known issue with some graph-grammar based compositions.

- RQ1: Does the composition of AO models produce a higher rate of composition conflicts than non-AO models?
- RQ2: What is the impact of AO modeling on the way conflicts propagate in the output model?

Our first null hypothesis assumes that the composition effort for combining AO models is essentially the same or even worse than for combining non-AO models (RQ1). Based on the fact that aspects crosscut many elements in a model (Section 2.2), the alternative hypothesis states that composing AO models leads to less conflicts than non-AO models. This would lead to the following null and alternative hypotheses, which will be analyzed for each conflict type (Section 2.4):

> **Null Hypothesis 1, $H_{1\text{-}0}$:** There is no difference between the conflict rates produced by the composition of either AO or non-AO models, or AO models lead to higher conflict rate.
> $H_{1\text{-}0}$: Rate (AO) ≥ Rate (non-AO).
>
> **Alternative Hypothesis 1, $H_{1\text{-}1}$:** Aspect-oriented modeling leads to a lower rate of conflicts than non-aspect-oriented modeling.
> $H_{1\text{-}1}$: Rate (AO) < Rate (non-AO).

Conflicts have a tendency to propagate in a composed model. That is, the introduction of one conflict can often lead to multiple other conflicts as a result of a "knock-on" effect. An example would be the conflict whereby a composed component is missing an important operation. This semantic conflict leads to a "knock-on" syntactic conflict if another component requires the operation. In the worst case, there may be long chains of conflicts all derived from a single conflict. Studying such propagation effects is important because propagation directly affects the effort in resolving conflicts—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 are interested in understanding the possible conflict propagation patterns in AO and non-AO models (RQ2). Similarly to the previous hypothesis, it is assumed that conflicts equally spread through output (non-)AO models. This leads to the second null hypothesis and alternative hypothesis as follows:

> **Null Hypothesis 2, $H_{2\text{-}0}$:** There is no difference in the way composition conflicts are propagated in AO and non-AO models, or AO models lead to higher propagation. $H_{2\text{-}0}$: Prop (AO) ≥ Prop (non-AO).
>
> **Alternative Hypothesis 2, $H_{2\text{-}1}$:** The use of aspects leads to a lower propagation of composition conflicts.
> $H_{2\text{-}1}$: Prop(AO) < Prop(non-AO).

In order to test the four hypotheses, a metrics suite was used in order to quantify both types of composition conflicts. These metrics are presented in Section 3.3. The metrics were applied to both non-AO and AO component models of an evolving software product line, described in the next section.

3.2 Case Study: Evolving an SPL

As previously discussed, model composition can be applied in different contexts and with different purposes (Section 2.1). We have selected a particular scenario to test our study hypotheses: the use of model composition to express the evolution of software product line (SPL) architecture.

Model Composition for Expressing SPL Evolution. Model compositions were defined to generate the new releases of the SPL architecture model. That is, the composition algorithms (Section 2.3) were used to define how each architecture model (M_A) of a SPL release and the new model increments (M_B) were going to be combined in order to generate the new architecture SPL release. The first input model, M_A, represents the current architecture of a SPL release, while the second input model, M_B, represents the delta capturing the modifications to M_A. The output model, M_{AB}, generated by the application of the composition algorithm, represents the next SPL release.

MobileMedia: the Target SPL. A product line, called MobileMedia [5], of 6 kLOC was selected to be the target case of the evaluation. The purpose of MobileMedia is to provide support for the manipulation of photos, music, and videos on mobile devices. A fine-grained description about its characteristics and how the evolution of Mobile Media happened can be found at [5][32][30]. The reasons for selecting this system in the evaluation are as follows. First, many releases of MobileMedia's architecture design models were produced by its developers. Second, two versions of the same product line, and the respective architectural models, were available for our investigation: an AO version and a non-AO version. This is a fundamental requirement to test our hypotheses (Section 3.1). Third, the last release of the architectural design has more than one hundred modules, and its architectural models are the main artifact to reason about change requests and derive new products. Fourth, the architectural models were produced by the original developers without any of the model composition algorithms under assessment in mind, thereby avoiding any bias and entailing a more natural software development scenario. Fifth, the architectural models (M_A) and the increment models (M_B) were conceived with modularity and changeability as key drivers. Sixth, we had available a total of seven fully-documented evolution scenarios, which could be expressed with model compositions (examples are given later).

Finally, MobileMedia met a number of other equally-important requirements, such as: (i) proper documentation of the driving requirements; (ii) the system evolved for more than three years, and the more recent releases have more than 100 modules; (iii) different types of change were realized in each release, including refinements of the architecture style employed, (iv) the system has been successfully used in other studies involving empirical evaluation of OO and AO implementations [5], and (v) the original developers were available to help us with the production and analysis of the composed models and the intended models. As such, all these factors provided a solid foundation for our study.

3.3 Quantifying Conflict Resolution

Our evaluation is concerned with the most-time consuming element of heuristic model composition, i.e. conflict resolution (Section 2.1). Therefore, our goal is to quantify: (i) the number of conflicts, and (ii) the activities required to modify the output model in order to make it reach the intended model. The analysis basically relies on a conflict measure, called *conflict rate* (CR), to quantify the amount of composition conflicts (Section 2.4) divided by the total number of elements in the output model. That is, CR allows computing the density of composition conflicts in the output models. This metric makes it

possible to assess the difference between the conflict rate of non-AOM and AOM (H_1). It is important to point out that CR is defined from multiple conflict metrics, which can be found in [30].

However, we also quantify the number of operations performed to transform the composed model into the intended model. We compute the number of creations, removals, and modifications needed to produce the intended model. This computation represents an estimation of the model *recovery effort* (RE). After we collect the RE measure, we perform an inspection of the output model to check if there was any occurrence of conflict propagation. This enables us to check if the presence of aspects in the input models has any impact on the way composition conflicts are propagated (H_2).

In the case study, the metrics above were applied to the composed models to analyze the conflicts and their propagation in each new MobileMedia release. The collected measures are used to assess if the output model has conflicts after the composition algorithm is applied ($diff(M_{CM},M_{AB}) > 0$). Then, a set of removals, updates, and creations were performed to resolve the conflicts. In order to come up with a suitable characterization of measures and respective releases, we defined a basic formalism for the metrics space of composition effort as follows.

A metric space is a set *M* equipped with a real-valued function CE(w,s) defined for all w,s ∈ M. Let M = {$R_{i,x,y}$, i = 1,…,n; x = override, merge; y = left, right}, where:

- *n* is a finite natural number representing the model release;
- *left* and *right* represent the direction of the composition relationship in the override algorithm.

For example, $R_{3,merge,right}$ represents the Release 3 that was produced by merging: Release 2 $+_{merge}$ Δ(Release 2, Release 3) → Release 3. Δ(Release 2, Release 3) represents the model elements that should be merged with Release 2 to transform it into Release 3. In practical terms, Δ represents the evolution to be inserted into the previous release. On the other hand, $R_{3,merge,left}$ would be: Δ(Release 2,Release3) $+_{merge}$ Release 2 → Release 3 (the inverse order can also be represented with an asterisk). So, the reader should note that the order of override-based composition can produce different output composite models [18]. Each model of a $R_{i,x,y}$ can be characterized by observing its syntactical and semantic properties. If we have a high conflict rate (CR) in an evolution scenario, then this implies a higher effort to resolve conflicts.

3.4 Evaluation Procedures

Once the case study was selected (Section 3.2) and the conflict resolution metrics were defined (Section 3.3), we needed to undergo a number of specific evaluation procedures. They are discussed in the following.

3.4.1 Target Model Versions and Releases

We have used both non-AO and AO versions of the MobileMedia models in order to test the study hypotheses (Section 3.1). These two model versions of the same system enabled us to identify if the presence of aspects in the input models had positive or negative effects on the quality of the output model.

Deriving AO and non-AO Model Releases. For each release of MobileMedia, we have applied each of the composition algorithms described in Section 2.3. That is, we have used the merge algorithm to compose two input AO models in order to produce a new AO release model; similarly, we applied the merge strategy to compose two input non-AO models in order to produce the next non-AO release model. We performed similar compositions with override and union algorithms. The goal was to identify if the outcomes, in terms of conflict rate and propagation (hypotheses), were the same or different. All the releases of the non-AO and AO versions realized exactly the same SPL features and variability points. They also underwent the same evolution scenarios, ranging from changes in heterogeneous mobile platforms and additions of many alternative and optional features [5]. Non-AO models were represented with conventional UML component models, while AO models were represented using the AOM language described in Section 2.2.

In fact, AOM (Section 2.2) is used in this work to represent the aspect-oriented model releases of the SPL under study. For example, in Figure 3, in addition to have interfaces (e.g., `PersistPhoto`), components (e.g., `ImageAcessor` and `AlbumData`), we also have aspectual components, such as the `ExceptionHandling` aspect. Moreover, we can also have some relationships: realization (e.g., between the components `BaseController` and `ControlPhoto`), dependency (e.g., between the component `NewAlbumScreen` and the interface `ControlPhoto`), and crosscutting (e.g., between the aspectual component `ExceptionHandling` and the component PersistPhoto, in which the service `loadAlbums(): void` is woven into the `ImageAcessor` component). The notation used in this work to express the architectural models has been used in other works [5] [26] and has shown to be effective for its purpose.

Model Releases and Composition Specification. We considered six releases of MobileMedia [5] in this study. They were selected because they were the ones where the changes implied visible modifications in the architectural design. For each new release, the previous release was modified in order to accommodate the features to be modified, inserted or removed. To implement a new evolution scenario, a model composition specification can remove, add, derive, or modify the entities present in the previous release. During the design of all releases, a main concern was to follow good practices of modeling.

3.4.2 Execution and Assessment Phases

The execution and assessment of the study were structured in three main steps, which are described in the following.

Model Refactoring Phase. The model refactoring is a pivotal activity to define the input models and, hence, to express the model evolution as an explicit model composition relationship. For this, MobileMedia's architectural models were initially refactored to specify the delta itself and to represent the change scenarios as composition relationships. To create the delta model it is necessary to identify the differences between the releases models and then gather them into the input model. To go about this, we took into account an evolution description

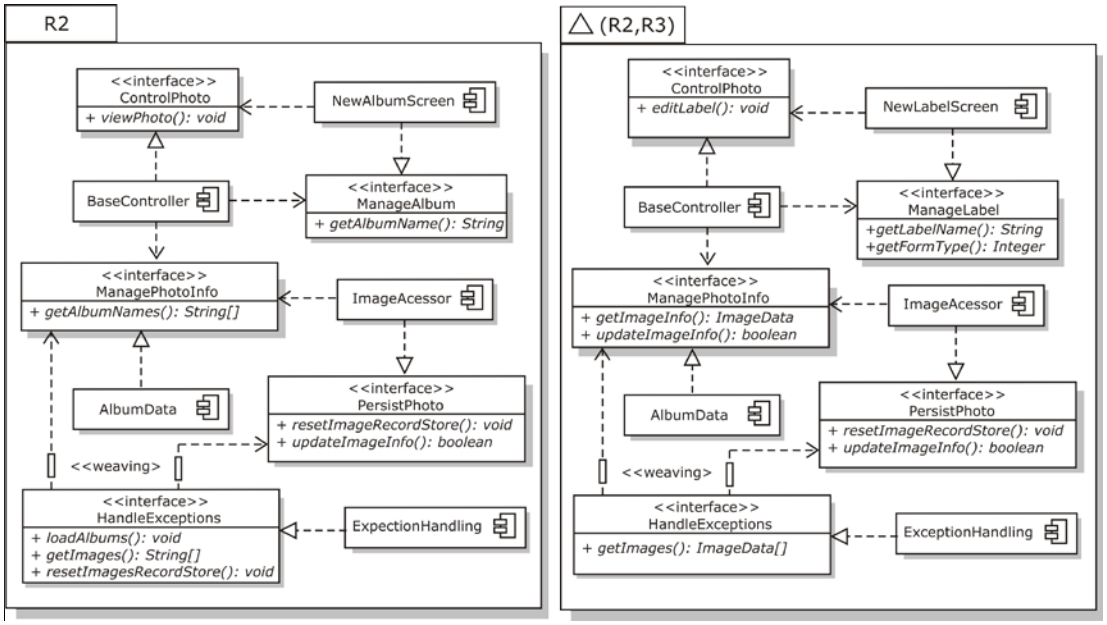

Figure 3. The input models: the base model (left) and delta model (right).

created by the original modelers involved in a previous study [5]. These descriptions specify in-depth the modifications needed to realize each evolution scenario (from one release to another). They allowed us to identify how the model elements were changed. For example, in the second evolution description, the Delta(R2,R3) were based on the description such as: the interface `ControlPhoto`—realized by `BaseController`—had the method `edilLabel(): void` added (see Figure 3).

Another example would be the change concerning the name of the interface `ManageLabel` to `ManageAlbum`. Thus, all model elements of the Delta(R2,R3) are derived from one evolution description, which ensures that the input model specification is free of bias. All input models and model evolution descriptions can be found at the study site [30].

Composition and Measurement Phase. From one release to another, 6 compositions were produced: 3 compositions following override, merge, and union from the current release to Δ, and 3 compositions in the inverse direction. We considered 5 evolution scenarios for the non-AO version as well as the AO version of the Mobile Media, totaling 60 compositions. The result of this phase was a document of composition descriptions, including the gathered data from the application of our metrics suite. Figure 3 presents partial input models being used in one of the releases, while Figures 4 (both sides) and 5 (right side) represent examples of composition based on merge, override and union, respectively. Figure 5 (left side) is the intended result of the composition (or mental model).

As well-validated metrics for model composition are not available yet, we used a suite of conflict metrics defined in our previous work [22]. The conflicts (and their effects) were identified manually using such conflict metrics. The identification of the conflicts was performed in 5 review cycles in order to avoid false positives/negatives. We also consulted the MobileMedia developers (Section 3.2) when needed, such

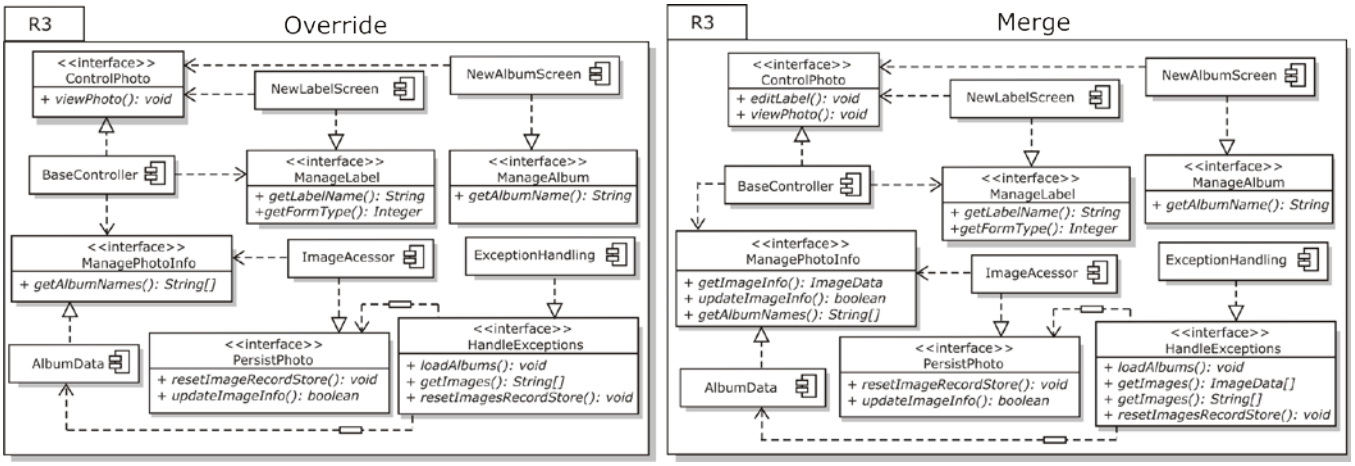

Figure 3. Output models produced by override (left) and merge (right) algorithms.

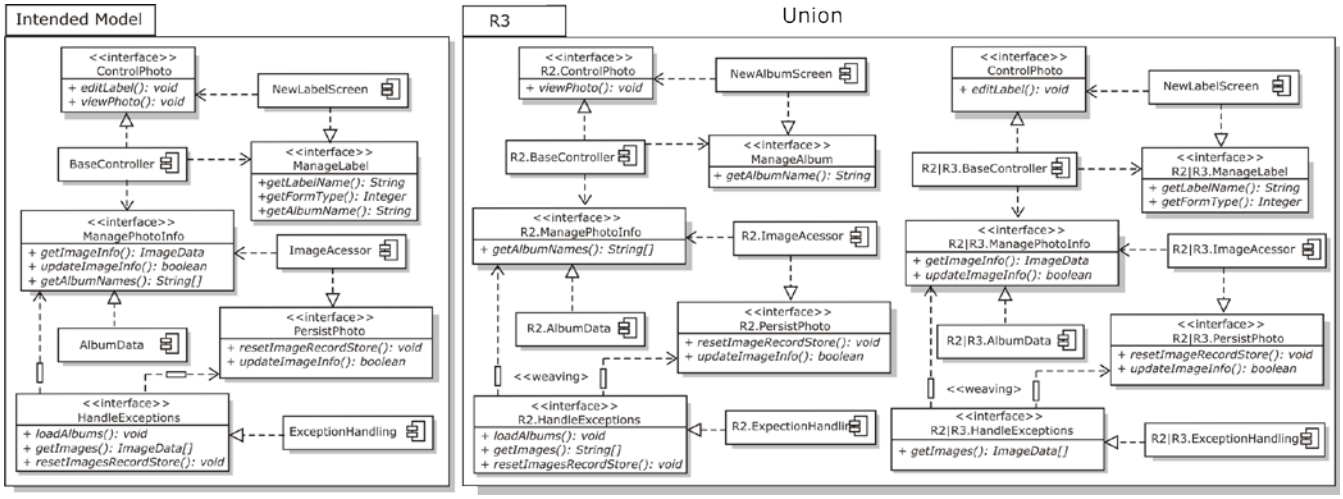

Figure 4. The intended model from the composition defined in Figure 3 (left).

The output model produced following the union algorithm (right).

as: checking and confirming specific cases of semantic conflicts. Even though the metrics have not been extensively validated, their feasibility and efficacy was also observed in a previous work involving other target applications [22].

Effort Assessment Phase. The goal of the third phase was to assess the effort to resolve the conflicts using the metrics described in Section 3.3. The composition algorithms were used to generate the evolved models, so that we could assess the impact of aspects 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 (Section 4.1) is concerned with pinpointing the conflict rates produced by composition of either non-AO or AO (H_1). The second stage aims at assessing the effort to resolve a set of previous identified conflicts and if the use of aspect has a higher impact on the way composition conflicts are propagated (H_2). We analyzed how conflict rate differs across the releases in order to detect potential benefits and drawbacks of using AOM in the input models. We have decided to focus the discussions on the merge and override algorithms, because the union algorithm did not present any additional interesting insights. However, all measurement results and the raw data are available at [30].

4. COMPOSITION EFFORT ANALYSIS

This section presents the results of applying the conflict resolution metrics (Section 3.3) to both the AO and non-AO output models realizing each SPL release (Section 3.4.1). Histograms are used to provide an overview of the data gathered in the measurement process. These histograms allow us to analyze the impact of aspects on model composition effort. Each histogram focuses on the application of a particular composition algorithm (Section 2.3). The Y-axis presents the values gathered for a particular metrics. The X-axis specifies the evolution scenarios.

Note that each pair of bars is attached to a pair of values, with the first capturing the performance of the AO version and the second capturing the non-AO one. Lower the value, the better is the performance of the modeling approach used. It is important to highlight that the results shown in the histograms were gathered with respect to the entire model. Based on the conflicts identified by the conflict rate (CR) metric, Section 4.1 discusses the findings related to the first hypothesis (H1). Section 4.2 relies on the metric for quantifying model recovery effort in order to support the analysis of the second hypothesis (H2).

4.1 H1: Aspects and Conflict Rate

Figure 6 illustrates the results for the CR metric obtained following the override algorithm. Figure 7 shows the results of the same metric for the merge algorithm. The first observation allows us to conclude that the conflict rate measures have favored aspect-orientation in both merge and override cases and for most of the evolution scenarios. This implies that the tally of conflicts to some extent is decreased whenever aspects are present in the models to-be-composed. The presence of aspects in the input models produced lower conflict rate than aspect-free models when the override algorithm is applied in both directions (right and left (represented by the *-columns)). For example, the conflict rate decreases from 1.72 (non-AO version) to 1.33 (AO version) in Scenario 2, which represents a reduction of 22.6% in favor of aspect-orientation. Similarly, the conflict rate decreases from 0.59 to 0.41 when the composition is performed in the left direction, which represents a reduction of 30%.

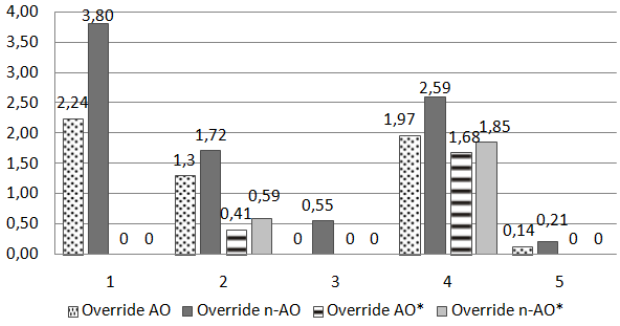

Figure 5. Conflict rate produced by the override algorithm

Moreover, it is well known that the greater the number of model elements that take part in compositions, the greater the likelihood of conflicts being generated. Nevertheless, the AO versions still had lower absolute measures of conflicts. For example, the absolute measure decrease from 38 (non-AO version) to 36 (AO version) in Scenario 2, which represents a reduction of 5.2% in favor of aspect-orientation. Similarly, the conflict rate decreases from 13 to 11 in the inverse order, which represents a reduction of 15.3%. The only case where aspect-free models led to a close CR was the application of the merge algorithm in the second release (Figure 7); this special case is discussed in Section 4.1.1.

The main reason for the superiority of the AO models is that changes, reified by the Δ model (Section 3.3), tend to be confined in fewer modules due to the superior modularization of crosscutting features in AO models. The confinement of modifications to aspects, in turn, leads to a better localization of both syntactic and semantic conflicts, thereby making them easier to detect and address in the output models. Therefore, we refute the null hypothesis $\boldsymbol{H_{1\text{-}0}}$ and confirmed the alternative hypothesis $\boldsymbol{H_{1\text{-}1}}$.

We have noticed that the decrease of conflicts observed in the AO models is potentially influenced by two factors: (i) *quantification*—the higher the quantification of aspects in input models, the higher the CR measures, and (ii) *obliviousness*—the higher the degree of obliviousness, the lower the CR measures in the output models (Section 4.1.1). Another predominant factor on the emergence of high conflict rates was the nature of the change. Independently of the degree of obliviousness and quantification in AO models, the nature of the change directly affected the conflict rate observed in the output models (Section 4.1.2). In the following, we elaborate these issues further and discuss examples that support each of these findings.

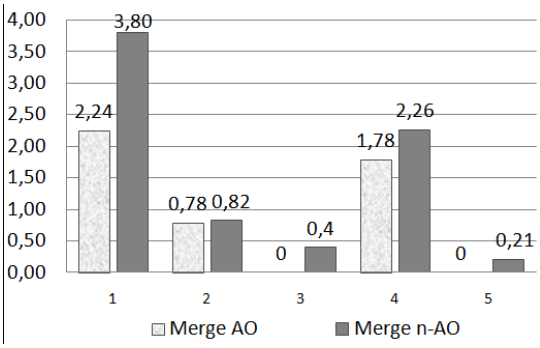

Figure 6. Conflict rate produced by the merge algorithm

4.1.1 Obliviousness and Quantification

We have observed that quantification influenced the CR measures. The presence of aspects with lower quantification (in the input models) led to fewer syntactic and semantic conflicts in the output models. When aspects were being used, for example, to encapsulate domain–specific features, a lower rate of conflicts manifested in the output models. On the other hand, we also observed that when a conflict arises in aspects with higher quantification (in the input models), higher rates of syntactic and semantic conflicts occurred in the output models. Therefore, the quantification mechanism may (or may not) improve CR results.

This category of aspects is the case where the aspects work as glue between a few elements in the base model and the changes realized by the Δ model. Aspects with a higher degree of quantification, such as exception handling (Figures 3, 4 and 5), affect the input base model in many places (join points). This was exactly the case in Scenario 2, where the non-AO version (CR = 0.82) has a measure close to the AO version (CR = 0.78) (Figure 7). Higher quantification increases the aspect scope and, therefore, the likelihood of aspects interfering with each other. When the merge algorithm was applied, the exception handling aspect (Figure 7) led to undesired superimpositions with other aspectual behaviors advising the same join points.

The overall rate of conflicts (CR measure) was usually lower in the AO version because most of the aspects were not affecting more than three elements. By overall rate, we mean the average of conflicts considering all the model elements. However, a careful analysis of the number of conflicts in individual model elements (e.g., a particular component) reveals some interesting information. The composition output of AO models consistently caused an increase on the number of conflicts for some specific model elements. For example, this can be observed in Scenario 4, when the highest number of conflicts emerged in both non-AO and AO versions. Despite the significant CR difference favoring the AO version, the component `BaseController` presented an increase (CR = 38) in relation to `BaseController` of the non-AO version (CR = 24). We noted that this problem occurred in situations where the components were affected by two aspects or more in the Δ model. In other words, when a base component had a high density of join points shared by multiple aspects, it generated a higher number of conflicts.

An additional interesting finding was that the composition of AO models tended to manifest fewer conflicts when the obliviousness degree of the base elements was higher. We have noted that the creation of new aspects (via the Δ model) for encapsulating new features implies that the modules in the input base model are more oblivious to the modification being implemented in the release. This observation holds for both mandatory and varying (optional or alternative) features. As a consequence, the combination of the AO modules tended to ripple fewer conflicts in the output models.

This finding implies that the presence of obliviousness is a good indicator that the model composition at hand will better adhere to the Open-Closed principle [31]. This principle states that "software should be open for extensions, but closed for modification". AO modeling conformed more closely to this principle in scenarios where the behavior in the new aspect (part of the Δ model) is more independent of the affected elements in the base model. This finding is illustrated, for instance, by Release 3. For instance, the `AlbumData` component demanded modifications in the non-AO version of Release 3 in order to include the feature of sorting photos by highest viewing frequency. On the other hand, the AO counterpart required no modification in this component. The reason was that the feature was modularly implemented by new components and the `PhotoSorting` aspect in the Δ model.

The open-closed principle was more closely adhered by the composition of AO models than non-AO models. However, this observation did not occur in all the cases. In general, this principle was fully achieved only when the Δ model was basically adding new elements to the base models. The other types of changes realized by the Δ model exerted more specific implications in the rate of conflicts detected in the output models. This issue is discussed in the following section.

4.1.2 The Effect of the Change Category

A careful analysis of the results has pointed out that the conflict rate is strictly affected by the category of changes to be applied to the base model. We identified four types of changes throughout our target SPL study:

- *Addition:* new model elements are inserted into base model; for instance, the new method `getFormType()` is inserted into the provided interface, named `ManageLabel`, of the component `NewLabelScreen` (Figure 4).
- *Removal*: a model element in the base model is removed; for example, the required interface `ControlPhoto` of the component `AlbumListScreen` is removed in the fourth MobileMedia release;
- *Modification*: a model element has some properties modified; for instance, the component `NewAlbumScreen` (Release 1) has its name modified to `NewLabelScreen` in Release 2 (Figure 4).
- *Derivation*: model elements are refined and/or move to accommodate the changes; for example, the provided interface `ControlPhoto` (with 14 methods) of the component `BaseController` (Release 3) has some methods moved to the provided interface `ControlPhoto` of the component `PhotoController` (Release 4).

Additions. As previously discussed (Section 4.1.1), the use of aspects has contributed to produce output model with much lower conflict rate when the evolution scenarios where dominated by *additions*. This finding is supported by the low conflict rate in Scenarios 3 and 5. The main reason is that the created aspects (in the Δ model) modularize the changes and insert them into the target model elements, without requiring their modifications. In these cases, we also observed that lower CR measures were observed in the AO models when the override algorithm is used and performed in the left direction. For all the other compositions, the conflict rate of the AO releases was equal or lower than the non-AO releases.

A concrete example of the superiority of the AO version was the decrease of the conflict rate from 3.8 to 2.24 in Scenario 1. This was due to the aspectual component, included in this release (via the Δ model), which advises 9 methods: (i) three of them in the interface `ManagePhotoInfo` of the component `AlbumData`; and (ii) 6 of them in the interface `PersistPhoto` of the `ImageAcessor`. This led to a CR decrease in the interface `PersistPhoto` from 11 (non-AO version) to 4 (AO version). In the same way, the `ManagePhotoInfo` had its conflict rate decreased from 9 to 6.

Modifications, Removals and Derivations. We could not find a recurring CR pattern (in favor of AO or non-AO versions) when modification was being realized. The AO version performed better in certain cases, while the non-AO version was better in others. On the other hand, the conflict rate was slightly higher in non-AO models when removals and derivations were applied. We also observed that a very high conflict rate occurred simultaneously in both AO and non-AO models when the change scenario was complex. This was the case when the change scenario involved a blend of modifications, removals and derivations. More specifically, this occurred in Scenario 4, when there is a significant architectural change: a single controller, for instance, was restructured as a set of specialized controllers.

Therefore, the heuristic composition algorithms were inefficient in widely-scoped architecture evolution, such as the refinement of the MVC (Model-View-Controller) architecture style of MobileMedia. This is also due in part to the name-based model comparison, which is not able to recognize more intricate equivalence relationships between the model elements. This comparison strategy is very restrictive whenever there is a 1:N correspondence relationship between elements in the two input models. An example of the 1:N relationship category encompassed the required interface `ControlPhoto` (Release 3) of the `AlbumListScreen` component. This interface was decomposed into two new required interfaces `ControlAlbum` and `ControlPhotoList` (Release 4), thereby characterizing a 1:2 relationship. For this particular case, the name-based model comparison should be able to "recognize" that `ControlAlbum` and `ControlPhotoList` are equivalent to `ControlPhoto`. However, in the output model (Release 4), the `AlbumListScreen` component provides duplicated services to the environment giving arise to an inconsistency. However, even in these cases the aspect orientation presented a lower conflict rate (e.g., see Scenario 4 in Figures 6 and 7).

It is known that a higher number of model elements may lead to a higher conflict rate when the composition is put in practice. But this was not the case with aspect-orientation. For instance, let's consider the fourth scenario. Although fewer composed elements (25) were observed in the non-AO version, the latter presents a higher CR measure (2.59). On the other hand, the AO version has a higher number of compositions (27), but the conflict rate is lower (CR = 1.97). A real example would be the `PhotoViewScreen` component, which decreased the number of conflicts from 3 (non-AO version) to 1 (AO version).

4.2 H2: Aspects and Conflict Propagation

We focus our discussion about conflict propagation on the analysis of model recovery effort, the RE measure (Section 3.3). This RE measure is a useful indicator to support the analysis of the presence (or absence) of conflict propagation (H2) in both AO and non-AO models. The higher the effort for recovering the output model (towards the intended composed model), the higher the chance of conflict propagation being observed in the output model. Figure 8 depicts the recovery effort measures to transform the output model produced by the override algorithm into the intended model. Similarly, Figure 9 shows the results of the same metric for the merge algorithm. The structure of the histograms follows those in the previous section.

We have concluded that aspects indeed affect the way conflicts flow throughout the output models. We identified a number of recurring conflicts in the AO models, which did not occur in the

non-AO models. In general, these conflicts specific to aspect orientation were caused by a conflict (or several) arising at a single aspect and spreading through all the affected elements in the base model. Therefore, we have found that there is a sensible difference on the way composition conflicts are propagated in non-AO and AO models. Therefore, we refute the null hypothesis $H_{2\text{-}0}$ and confirm the alternative hypothesis $H_{2\text{-}1}$.

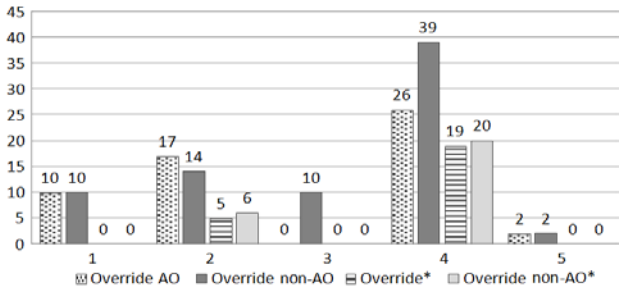

Figure 7. Conflict resolution effort to recover the output model produced by override algorithm.

4.2.1 Quantification and Model Recovery Effort

According to previous discussion (Section 4.1.1), aspects with higher quantification contribute to higher conflict rates in AO models. An inspection of the output models, however, pointed out that this problem occurred because these aspects led to higher conflict propagation manifesting during the model composition process. Surprisingly, the higher rate of conflicts in such AO models does not reflect in more effort to recover the output models and produce the intended composed model. In other words, an interesting finding is that a high degree of quantification does not lead to more effort to recover the output model. The RE measure often tends to be similar in AO and non-AO models.

This phenomenon can be illustrated, for example, in Scenario 2 (Figure 7), where the AO version presents a higher conflict rate (CR = 1.37) than the non-AO version (CR = 0.82). However, the model recovery effort is equal to 9 for both AO and non-AO versions (Figure 9). This was the case of conflicts arising at a reusable exception handling aspect (modified by the Δ model). When conflicts arose in such an aspect, they rippled through all the model elements directly advised by the aspect. During the model recovery process, there was a need to fix only the conflict in the specification of the exception handling aspect.

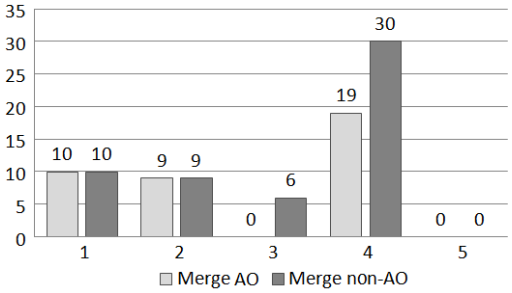

Figure 8. Composition effort to recover the output model produced by merge algorithm

Therefore, although AO and non-AO versions present different conflict rates in certain evolution scenarios (e.g., Scenario 1 in Figure 7), the effort to recover the output model from the conflicts in both versions is similar. The effort directly depends on how instances of conflicts are interrelated. Propagation channels of conflicts were more common in AO models as discussed above. For example, despite aspect orientation exhibiting a conflict rate close to the non-AO conflict rate in Scenario 2 (Figure 7), the conflict resolution effort is similar to non-AO models. Thus, when the conflict that is responsible for propagation is identified and resolved, all conflicts are indirectly resolved as well.

4.2.2 Shared Join Points and Cyclic Propagation

We have noticed that when a conflict emerged in a highly-coupled base module (e.g., a controller in MobileMedia), it led to a higher degree of conflict propagation in the AO versions than the non-AO versions. This problem was particularly observed when the highly-coupled base module was the source of join point shadows shared by multiple aspects. For instance, we have analyzed the conflict channels triggered by a conflict arising in the `BaseController`, a central model element in the Mobile Media architecture. We observed that the conflict propagation affected four components in the non-AO version, namely `AlbumListScreen`, `PhotoListScreen`, `PhotoView Screen`, and `AddPhotoToAlbumScreen`. However, the propagation affected three additional modules (aspects) in the AO version.

The `HandleExceptions` interface had a method signature modified from `String[] getImages(String record-Name)` to `ImageData[] getImages(String record-Name)`. However, the `R1.HandleExceptions` incorrectly overrides `Δ(R1,R2).HandleExceptions`. As a result, this method was incorrectly present into the output model, which gives arise to some functionality conflicts. This propagation was spread through the component `AlbumData`, because the aspect is no longer able to introduce the expected method `ImageData[] getImages(String record Name)` into the provided interface `ManagePhotoInfo` of `AlbumData`. As a consequence, `AlbumData` does not provide some expected services to the environment. Hence, conflicts are propagated through the component `BaseController` and `ImageAcessor`.

It is interesting to note that `ImageAcessor` is also affected by a conflict that emerged from `AlbumData`. As `ImageAcessor` requires the service (`ImageData[] getImages(…)`) provided by the interface `ManagePhotoInterface`, it is not able to correctly provide the all services defined in the provided interface `PersistPhoto`. Hence, the `AlbumData` is also re-affected by a conflict that previously arose from it. This phenomenon represents cyclic conflict propagation. On the other hand, this propagation is solved in the composition $R_{2,overide,left}$ due to the `Δ(R1,R2).HandleExceptions` override the `R1.HandleExceptions`, decreasing the conflict rate from 1.3 in $R_{2,overide,right}$ to 0.41 in $R_{2,overide,left}$.

5. RELATED WORK

Model composition is a very active research field in many domains, including database integration [3], composition of web services [15], merging of statecharts [2], model composition in product lines [1], composition of UML models [18][19][10] [12], aspect-oriented modeling [13][11][14], and AO composition of models [8]. However, there is little related work

focusing on the quantitative and qualitative assessment of AOM. In general, most of the research on the interplay of AOM and 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 models in general [7][16][17]. 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.

More specifically, to the best of our knowledge, researchers have neglected the assessment of how aspects impact model composition effort. The need for assessing models during a model composition process has neither been pointed out nor proposed by current model composition techniques [9][2][8] [18]. For example, the UML built-in composition mechanism, namely package merge [18][19], does not define metrics or criteria to assess the merged UML models. Moreover, it has been found to be incomplete, ambiguous and inconsistent [19].

The lack of quantitative and qualitative indicators for model compositions hinder the understanding of side effects peculiar to certain model composition strategies (in the presence of aspects or not). 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 an extent. However, as researchers' focus has shifted to the activities related to model management (such as model composition, evolution and transformation), the shortcomings and limitation of UML model metrics have become more apparent. Some authors [1][12][13][14][15][16][17][18] have proposed a set of metrics that consider UML models' properties. These works have shown that their measures satisfy some properties expected for good measures of design models. However, these metrics cannot be employed to assess problems that may arise in a model composition process such as semantic conflicts.

There are some specific metrics available in the literature for supporting the evaluation of model composition specifications. For instance, Chitchyan and colleagues [20] have defined some metrics to quantify the effort to specific compositions between two or more requirements models, such as scaffolding and mobility. However, their metrics are targeted at evaluating the reusability and stability of explicit model composition specifications. Boucké and colleagues [25] also propose a number of metrics for evaluating the complexity and reuse of architectural model compositions. However, in this paper, we have focused on the evaluation of heuristic composition algorithms, such as merge and override, where explicit model compositions are not provided upfront. In addition, we have focused on analyzing the impact of aspects on the effort to resolve emerging conflicts in output models. Therefore, existing metrics (such as those described in [20][25][23]) cannot be directly applied to our context.

6. THREATS TO VALIDITY

Our exploratory study obviously has a number of threats to validity that range from: (i) the use of single target application and a single AOM language, to (ii) the use of specific metrics to compute the conflict resolution effort. Obviously, more investigations involving other case studies with compositions of larger UML models are required. We observed that the number of properties and details (i.e., granularity) of the model elements taken into consideration throughout the compositions affect directly the composition results. Consequently, it is necessary to observe that, to generalize our findings, other types of model with different levels of abstraction are needed to make further investigation.

Further empirical evaluations are indeed fundamental to confirm or refute our findings in other real-world design settings involving UML model compositions. However, it was never our goal to conduct a controlled study. Our investigation represents a first stepping stone, where a number of initial findings can be used to drive the experimental designs of more controlled studies in the future.

7. CONCLUSIONS AND FUTURE WORK

Model composition is one of the pillars of AOM, and it is an operation intended to be used in many software development activities. Hence, software designers naturally become concerned about the quality of the composed models. This paper represents a first exploratory study to assess the potential advantage of aspect-orientation in reducing conflict resolution effort. In our study, model composition was used to express the evolution of architectural models along six releases of a software product line. Three canonical algorithms for heuristic model composition have been applied, and two of them were discussed in detail in this paper.

As expected, we found that the presence of aspects in input models improved modularization and, therefore, tended to better localize conflicts. We have also observed: (i) a higher degree of obliviousness between base models and aspects led to a significant decrease of conflicts when compared with the non-AO model counterparts, and (ii) aspects with higher quantification were the cause of higher conflict rates in AO models. Another interesting finding was that, even in scenarios where conflict rate of AO models was close to (or higher than) the conflict rate of non-AO models, conflict resolution effort was similar in AO and non-AO models. This means that the time spent for recovering output AO models from emerging conflicts is, at least, similar to non-AO models. All these findings were independent of the specific composition algorithms being assessed. These results provide some initial indication that aspect-orientation may alleviate conflict resolution effort.

We should point out that assessing the added value of AOM in model composition is in its initial stage and there is very little experience that can be used to determine the feasibility of current approaches. This study represents a first exploratory study that investigates the impact of aspects on conflict resolution effort. However, further empirical studies are still required to evaluate the impact of AO modeling on model composition in real-world settings. We also need to better understand if aspect orientation provides some gain or not: (i) when applied to 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 future under different circumstances.

Acknowledgments: this work was funded by Faperj – distinguished scientist grant (number E-26/102.211/2009); by CNPq – productivity grant (number 305526/2009-0), Universal project (grant number 483882/2009-7), and PhD scholarship (number 142857/2009-2); and by PUC-Rio – productivity grant.

8. REFERENCES

[1] P. Jayaraman, J. Whittle, A. Elkhodary, and H. Gomaa. Model Composition in Product Lines and Feature Interaction Detection using Critical Pair Analysis. In *MODELS'07*, pages 151–165, 2007.

[2] S. Nejati et al. Matching and Merging of Statecharts Specifications. In: *ICSE'07*, pages 54–64, 2007.

[3] P. Bernstein and S. Melnik. Model Management 2.0: Manipulating Richer Mappings. In: *SIGMOD'07,* pages 1–12, ACM Press, 2007.

[4] V. Basili et. al. The Goal Question Metric Paradigm. In: *The Encyclopedia of Software Engineering, Vol. 2*, pp. 528–532, John Wiley and Sons, 1994.

[5] E. Figueiredo et al. Evolving Software Product Lines with Aspects: An Empirical Study on Design Stability. In: *ICSE'08*, pages 261–270, 2008

[6] S. Clarke and R. Walker. Composition Patterns: an Approach to Designing Reusable Aspects. In: *ICSE'01*, pages 5–14, 2001.

[7] R. France and B. Rumpe. Model-Driven Development of Complex Software: A Research Roadmap. In: *Future of Software Engineering* at ICSE'07, pages 37–54, 2007.

[8] Y. Reddy et al. Directives for Composing Aspect-Oriented Design Class Models. *LNCS Transactions on Aspect-Oriented Software Development*, 1(1):75–105, 2006.

[9] T. Cottenier, A. Berg, and T. Elrad. The Motorola WEAVR: Model Weaving in a Large Industrial Context, In: *AOSD'07*, 2007.

[10] R. France, D. Kim, S. Ghosh, and E. Song. A UML-Based Pattern Specification Technique. *IEEE Trans. Software Eng.*, 30(3):193–206, 2004.

[11] R. France, I. Ray, G. Georg, and S. Ghosh. Aspect-Oriented Approach to Early Design Modeling. *IEE Proceedings: Software*, 151(4):173–185, 2004.

[12] R. France, S. Ghosh, and T. Trong. Model Driven Development Using UML 2.0: Promises and Pitfalls, *Computer*, 39(2):59–66, 2006.

[13] S. Clarke and E. Banaissad. *Aspect-Oriented Analysis and Design: The Theme Approach*. Addison-Wesley, Upper Saddle River, 2005.

[14] E. Tempero and R. Biddle. Simulating Multiple Inheritance in Java. *Journal of Syst. and Soft.*, 55(1):87–100, 2000.

[15] N. Milanovic and M. Malek. Current Solutions for Web Service Composition. *IEEE Internet Computing*, 8(6):51–59, December 2004.

[16] C. Lange and M. Chaudron, Effects of Defects in UML Models: An Experimental Investigation. In: *ICSE'06*, pages 401–411, 2006.

[17] C. Lange, B. DuBois, M. Chaudron, and S. Demeyer. An Experimental Investigation of UML Modeling Conventions. In: *MoDELS'06*, pages 27–41, 2006.

[18] J. Dingel, Z. Diskin, and A. Zito, Understanding and Improving UML Package Merge. *Journal of Software and Systems Modeling*, 7(4):443–467, 2008.

[19] OMG. *Unified Modeling Language: Infrastructure version 2.2*. Object Management Group, February 2008.

[20] R. Chitchyan et al. Semantic vs. Syntactic Compositions in Aspect-Oriented Requirements Engineering: An Empirical Study. In: *AOSD'09*, pages 36–48, 2009.

[21] K. Farias, A. Garcia, and C. Lucena. On the Comparative Evaluation of Aspect-Oriented Model Composition Techniques. In: *Proc. of the III LA-WASP'09*, 2009.

[22] K. Oliveira, A. Garcia, and J. Whittle. On the Quantitative Assessment of Class Model Compositions: An Exploratory Study. In: *1st Wkshp. on ESMDE* at MODELS'08, 2008.

[23] J. Conejero, E. Figueiredo, A. Garcia, J. Hernández, and E. Jurado. Early Crosscutting Metrics as Predictors of Software Instability. In: *TOOLS-Europe'09*, pages 136 – 156, 2009.

[24] A. Molesini, A. Garcia, C. Chavez, and T. Batista. Stability Assessment of Aspect-Oriented Software Architectures: A Quantitative Study. *Journal of Systems and Software*, 2009.

[25] N. Boucké, D. Weyns, and T. Holvoet: Experiences with Theme/UML for Architectural Design in Multiagent Systems. In: *MASSA'06*, pages 87–110, 2006.

[26] A. Garcia et al. Representing Architectural Aspects with a Symmetric Approach. In: *EA'09* held at AOSD'09, 2009.

[27] T. Cotternier, A. van den Berg, and T. Elrad. Modeling Aspect-Oriented Composition. In: *Int. Workshop on AOM* at MODELS'05, 2005.

[28] J. Whittle and P. Jayaraman, MATA: A Tool for Aspect-Oriented Modeling Based on Graph Transformation. In: *Int. Workshop on AOM MoDELS*, pages 16–27, 2007.

[29] Kompose: A Generic Model Composition Tool. http://www.kermeta.org/kompose, 2009.

[30] Assessing the Impact of Aspect on Model Composition Effort, http://www.inf.puc-rio.br/~kfarias/aosd10

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

[32] Evolving Software Product Lines with Aspects. http://www.lancs.ac.uk/postgrad/figueire/spl/icse08/

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