

# A reference process for model composition

Cédric Jeanneret  
EPFL  
Lausanne, Suisse  
cedric.jeanneret@epfl.ch

Robert France  
Colorado State University  
Fort Collins, USA  
france@cs.colostate.edu

Benoit Baudry  
IRISA  
Rennes, France  
bbaudry@irisa.fr

## ABSTRACT

In a Model Driven Engineering (MDE) environment, composing several models to produce a single integrated model is an important model management activity. The complex structure of models makes manual model composition a difficult and tedious task. This problem has given rise to several proposed approaches automating model composition. In this paper, we propose a process framework for model composition that can be used to compare different composition approaches. One of the key insights provided by the framework is that model composition is not an operator that can be completely automated.

## Categories and Subject Descriptors

I.6.5 [Computing Methodologies]: Simulation and Modeling—*Model Development*

## General Terms

Model Composition

## Keywords

Model, Composition, Process

## 1. INTRODUCTION

Model Driven Engineering is a paradigm devoted to tackling growing software complexity: the gap between the problem description and the solution design is reduced by systematically transform abstract models to implementation and deployment artifacts. In MDE, models are the primary development artifacts. Usually, a problem or a system is not modeled in a single model capturing all the properties. A system or the problem can be modeled from a set of viewpoints, and only properties relevant to a particular viewpoint are represented in the corresponding model. Furthermore, the modeling language used may vary across these different viewpoints. The view-oriented decomposition can

be guided by a methodology like 'Use case driven development' [8], 'Theme/UML' [4] or 'Aspect oriented modeling' [6]. Having a set of models, each one representing a single aspect (or use case), is suitable for analysis by specialists, but these models have to be composed to obtain a single, integrated model. This integrated model can be used to feed a code generator, to assess the consistency among the models or to uncover unwanted emergent properties (resulting from the composition of two interfering concepts). Several approaches have been proposed to automate or to support model composition. There seems to be a lot in common with these approaches, but it is difficult to compare them because they use different technologies, they accept different modeling languages or they require different inputs from the users. Our work addresses this problem by providing a reference framework that makes this comparison possible.

Model composition has often been considered as an operator [1, 3], that could ideally be applied in all cases where models have to be composed, independently of their metamodel or their model kind. This would be possible if the operator could infer the intent of the models and the possible interactions across the models directly from the input models. Unfortunately, it is extremely difficult to infer these semantics concepts from the models' representation. There is a gap between the semantics of a model, that is the set of concepts held in the modeler's mind, and its syntactic form as defined in a given metamodel. The problem presented by this gap has already been identified and has given rise to a classification of model composition operators [7]. Even though most existing approaches claim to provide an operator for model composition, they are all implemented as a parametrized sequence of operations on some sets of objects. We believe that this sequence of operations is driven by a process that is often left implicit in work describing composition operators.

In this paper, we propose a process framework for model composition. We also explore the flexibility available in composition processes. We provide an analysis showing how existing approaches fit into the framework. For this purpose, we use Atlas Model Weaver [2], signature-based composition [10] and MATA [9].

This paper is organized as follows: section 2 presents the proposed process framework. Existing techniques are briefly presented in section 3, and these techniques are related to the process framework in section 4. We conclude this paper with a discussion on the implications of our contributions.

## 2. MODEL COMPOSITION PROCESS

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AOM Workshop AOSD'08, Brussels, Belgium  
Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

In this section, we propose a reference process framework for model composition. We start by analyzing the composition problem at the conceptual level, outlining the concerns related to model composition. We then propose a process that addresses these concerns at the syntactic level. Finally, we detail the variation points in the process framework, highlighting the flexibility a composition process can provide.

In order to compose two models, one has to address 3 concerns:

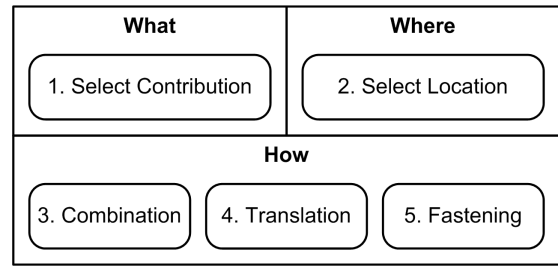
- What: Decide which concepts described in the models will be composed.
- Where: Select a *location* where the insertion or modification will take place in the destination model (the composed model).
- How: Determine how to integrate the selected concepts in the composed model to obtain the desired result.

In other words, composing models consists of selecting *what* to compose, *where* to place the composed concepts and finally determining *how* to integrate the concepts at the selected location in the composed model.

Even though composing models is simple and intuitive at the semantic level, things become complicated when we try to define the syntactic composition operators that realize the conceptual composition operations. Indeed, concepts can be represented by more than one model element or using different modeling languages. For instance, consider the concept of sales account. It is possible that in one model it is represented as a class in an UML class diagram, and in another as an entity in a ERD schema. Even if we consider two models written in the same language, discrepancies can appear between two representations of the same concept. Hence, the account's open date may have been encoded with a string by a modeler, while another modeler represented this exactly same concept as a date object. Unfortunately, since composition technologies must work at the syntactic level, these problems must be handled by syntactic composition operators.

In the framework, the composition process at the syntactic level is iterative, in the sense that a set of concepts are composed one after the other. Indeed, since we consider models as sets of objects, the order in which the composed model is built matters, because the location and the integration will reference elements that have already been composed (that means elements already present in the composed model). The process finishes when the composed model is considered complete, that is, when all the relevant elements have been processed. An iteration consists of the following sequence of actions:

1. Pick up some model elements from one or more models. These elements form the *contribution* of that iteration.
2. Select a place in the composed model where the insertion will take place. This particular place will be named *location*.
3. If more than one model contributes in this iteration (if there are more than one contributions in this iteration), the contributions will have to be combined into a single a set of model elements, namely a *fragment*.



**Figure 1: Mappings between high level concerns and the steps of our process**

4. When models conform to different metamodels, this fragment is then translated to another fragment, but this time conforming to the composed model's metamodel (which can be one of the inputs metamodel or a different one).
5. Once the fragment is inserted into its location, it may have to be *fastened* to its context. This action makes sure that the fragment is connected to the rest of the composed model.

The figure 1 illustrates the mapping between high level concerns and syntactic composition actions. The first step of the process deals with the *What* concern. By picking up related elements across the input models, the users will be able to combine them in a subsequent action. Once the contribution is chosen, the users have to select *where*, in the composed model, the insertion will take place. This is the purpose of the second step. The *How* concern is addressed by the three last steps of the process. In step 3, the users define how the elements are to be combined (this step is called Combination). The fourth step is taken if all the models (input and output) do not conform to the same metamodel. Elements must then be translated to the output metamodel. The users decide how to achieve this translation. Once the contribution is translated and inserted into its location in the composed model, the users can fasten this freshly introduced elements to the rest of the composed model. For example, a generalization relationship can be established between the class 'Sales Account', the contribution of a current iteration, and a class 'Account' contributed in a previous iteration.

This process must be flexible enough to address all model composition needs. This flexibility is conditioned by variations allowed at any of the steps, and on the ordering of the iteration and the stopping condition of the process. We describe flexibility opportunities in our process framework in the remainder of this section.

Contribution selection presents two degrees of freedom. First, the provided granularity: it can range from a single model element to a set of related elements. For example, given a sequence model, an approach can allow the user to pick up the call message, the behavior execution and the return message as a single contribution, whereas some other approach will restrict the user to the selection of a single element. The second degree of freedom concerns the liberty granted to the users regarding the elements they can select. Indeed, constraints can be imposed when elements are selected from more than one model. For example, if a class is selected in the first input model, a tool may force the users to pick up a class in the second input model.

The location (the place where the fragment will be inserted) offers a single variation point: it can be determined by the tool (fixed), resolved via a policy or selected by the user.

There are many ways to combine two contributions into a single fragment. This combination can be encoded in the composition tool, specified with predefined operators like merge or override (on a case by case basis or via a policy) or can be user-defined.

By their nature, translation and fastening only bring a single degree of freedom. Either they are available (in that case, one can reasonably expect that the provided mechanism is flexible enough to fit all kind of needs) or not.

To summarize, our process framework provides 6 points of variations that users can tune to make the composition process fit their particular needs: granularity and selectivity for the contribution concern, freedom for the location's choice, the expressiveness available for combination and the existence of translation or fastening mechanisms. In order to illustrate the validity of this framework and the pertinence of our variation points, we show how three composition approaches fit within our framework and identify the choices that have been made with regards to the flexibility. The next section will present a short introduction to these selected techniques. The alignment of these tools to our process framework is described in section 4.

### 3. EXISTING TECHNIQUES

Atlas Model Weaver (AMW) is a facility for establishing relationships between elements from different models. These links are captured in a weaving model; moreover this model conforms to a weaving metamodel, declaring the kind of relationship that can be modeled. The semantics of these relationship can be defined as a transformation expressing how related elements have to figure in the composed model. Once mappings between models (actually the weaving model) are established, a model transformation written in ATL can compose the woven models into a composed model. Note that links between models can either be established manually (via the AMW built-in editor) or inferred via a match operator (an ATL transformation taking two models as input and producing a weaving model), usually relying on some heuristics. More details on how AMW can be used for model composition can be found in [2]

Different than AMW at the first glance, but similar in principle, Kompose is a composition tool built on top of Kermet, a metamodeling environment allowing to add behavior to metamodels. Kompose is therefore a metamodel for model composition. However, in contrast to AMW where relationships are instantiated between model elements, Kompose relies on the use of signatures to infer these relationships. A signature is a set of values extracted from model elements' properties' values, such as its name and its type. If two model elements present the same signature, there are considered as matching, and will be merged in the composed model. Composition is indeed performed in two steps: first, elements' signatures from the actual containers are compared to each other, then if some signatures are matching, the two related elements are merged (and the algorithm recursively considers these elements as the actual containers) into a single element. Otherwise, elements without corresponding elements in the opposite input model are simply added to the container of the composed model. This ap-

proach is subject to two issues, namely conflicts (in the sense that two elements present the same signature, but differ on irrelevant properties) and misalignment between signature and the concept modeled by the element (the signatures match but should not, or, on the opposite do not match but should). To address these issues, Kompose provides the user with pre-merge directives, enabling him to alter the input models such that no more conflict is raised or models are aligned. Kompose can also remove undesired emerging properties or enforce the model to possess some properties by allowing the user to transform the output model with post-merge directives. Additional information on the Kompose metamodel and the way it can be extended to deal with a specific metamodel is available in [11, 5].

Modeling Aspects using a Transformation Approach (MATA) proposes a radically different approach than the two previous ones. Indeed, the composition becomes asymmetric, as one model plays the role of base model while the other is seen as an aspect. Even though the aspect model is designed using the same concrete syntax than the base model, it will be interpreted as a graph transformation to be applied on the base model. Hence, the aspect model is made of two overlapping parts: a pattern and a composition specification. Syntactically, aspect and base models will differ by the presence in the aspect model of pattern variables and annotations used for composition specification: create, delete and context stereotypes. Create stereotype annotate elements that will be added in the base model, whereas elements marked as delete will be removed from the base model. These annotations are automatically inherited by elements owned by the annotated element, removing from the user the burden to apply the stereotype on numerous elements. To prevent this optimization to be carried on, the user can use the context stereotype. The application of the aspect model to the base model is achieved in two phases. First, a match of the pattern has to be found in the base model and then, this match is modified accordingly to the composition specification. MATA has been described with full details in [9, 12]. Note that even if MATA was designed for aspect-oriented modeling, one can emulate model composition by applying successively several aspects.

In this section, we have briefly introduced 3 approaches for model composition. In the next section, we will show how they fit into our process framework.

### 4. ANALYSIS

To validate our framework, we fit 3 selected composition approaches into it. First, we analyze how each approach supports our process steps and then describe the flexibility these techniques provide.

The table 1 presents a summary of the analysis. Approaches are represented as columns whereas process steps are given in rows. Note that this table relies on our own interpretation of the analyzed solutions. However, it shows that the approaches fit, to some extent, to the framework.

Since AMW is based on ATL transformations that are declarative, the order in which the composed model is built is determined by the ATL engine. Furthermore, the composition stops as soon as no more ATL rules can be applied. Concerning the contribution selection, AMW allows the user to define relationships between two elements coming from distinct models. The manner with which these elements will be combined depends on the type of the weaving

**Table 1: The support of the composition process by existing approaches**

	AMW	Kompose	MATA
Ordering	ATL rules application order	Follows ownership relationships	Aspects application sequence
Stopping Criteria	no more ATL rule applies	All model elements processed / post-merge directives	Aspects application sequence
Contribution	Weaving model	Signatures / pre-merge directives	Aspect
Location	ATL transformation	Defined by the location of input elements	Pattern
Combination	Weaving metamodel / Weaving model	Merge / merge directives	Composition specification
Translation	ATL Transformation	N/A	N/A
Fastening	ATL Transformation	Post-merge directives	Composition specification

link relating them. These link types, defined in the weaving metamodel, specify combination operators that will be interpreted by the ATL transformation. Finally, the fastening and the translating phases can be dealt within the ATL transformation. Because there is no existing implementation for model composition with AMW and existing documentation does not mention this part of the approach, it is unclear where elements are inserted, but this location depends for sure on the ATL transformation that will compose models. To summarize, the what is defined by the weaving model, the how is encoded by the weaving metamodel and the where is determined by the ATL transformation.

Within Kompose, the order is determined by a recursive visit of input models (induced by the ownership relationships). The process stops as soon as all elements have been visited; however, the user can remove elements from the destination model via post-merge directives, undoing the work achieved by unwanted iterations. In this approach, matching pairs of elements are determined by signatures comparison. Only elements in matching containers are compared. Furthermore, users have the opportunity to alter elements before the composition. Such opportunities can be used to force or disallow matching. The manner in which matching elements are combined is fixed, unless overridden by a merge directive: the two elements will be represented as a single element in the composed model (remember that conflicts have to be fixed with pre-merge directives), and its content (the elements it owns) is the merged content of both elements. The place where fragments are inserted is precisely defined in Kompose: the element will be inserted in the container built out of the input elements' containers. Kompose uses the abstract syntax of models rather than the concrete syntax to determine location. It is not possible to choose the relative concrete location where an element is inserted in its container in cases where the relative order matters (e.g. in sequence diagrams). However, a specific fragment can be moved to another location with a post-merge directive. Kompose makes the assumption that the input models and the output model conform to the same metamodel, therefore no translation is required. The fastening phase is achieved with post-merge directives. Globally, Kompose has a fixed behavior in most of the steps of our process and this rigidity allows to compose model with minimal inputs (the two input models and signatures definition). This rigidity can be softened with the help of merging directives.

In MATA, the order and the stopping criteria of the composition process is defined by the sequence of aspects to be applied. Note that this sequence can be validated by a critical pair analysis, which detects structural dependencies among aspects. However, during the application of a particular aspect, the order and the stopping criteria are inferred by the transformation engine. A contribution is defined by the aspect itself, which is made of any number of elements. The aspect will be applied where the pattern matching mechanism will find candidates that match the specified pattern. Once a match is found, the composition specification defines how elements are combined and fastened, in other words, how the matching is modified. Since aspects are modeled with an extended version of the modeling language they are applied to, no translation mechanism is provided. Given that an aspect is made of a pattern and a composition specification, the particularity of MATA is that all the concerns of our process are addressed within the same input artifact, the aspect.

So far, we have shown how the different approaches implement the steps of our process, be it via some additional input or by a built-in mechanism. The table 2 illustrates how flexible the existing solutions are, according to the discussion provided in section 2.

The flexibility provided by AMW is distributed in two levels. First, a domain specialist designs the weaving metamodel and the ATL transformation that will interpret the weaving model. The flexibility provided at this level is the one of the ATL model transformation language, which is limitless. On the second level, when actually composing models, the users model the relationships between elements coming from the input model. These relationships form a model conforming to the previously designed weaving metamodel. Therefore, the users are granted a limited flexibility while deciding what two compose and how to compose it.

Kompose has the most restricted flexibility, if composition directives are not considered. The ordering of the iterations and the stopping criteria is dictated by the tool. Moreover, the granularity of contributions is limited to single model elements where as the selectivity is inferred by signatures (whose definitions are the only parametrization of this composition operator, ignoring directives). Similarly, the way elements are combined and the place where elements are inserted in the destination model is wired in the tool. However, composition directives provide the users with a possibility to

**Table 2: The flexibility provided by the existing solutions**

	AMW	Kompose	MATA
Ordering	Inferred by ATL engine	Fixed	User-defined / inferred by transformation engine
Stopping Criteria	Defined by ATL engine	Fixed (post-merge directives)	User-defined / inferred by transformation engine
Contribution Granularity	Model element	Model element	Arbitrary
Contribution Selectivity	Arbitrary	Limited to elements from matching containers / elements from the same meta-class / links inferred from signatures (pre-merge directives)	Arbitrary
Location Freedom	(Defined in ATL Transformation)	Fixed	Defined by pattern
Combination Expressiveness	User-defined operators	Fixed (merge directives)	Arbitrary
Translation Existence	Yes	No	No
Fastening Existence	Yes	Yes	Yes

override Kompose default behaviors.

As claimed by the team behind it, MATA is the most flexible approach according to our process. The users are granted more freedom when deciding what to compose and how to compose. Note that since the location is specified with a pattern, the exact place where the aspect will be applied is computed by the pattern matching mechanism, and it is therefore not under users' full control. MATA does not allow composing models conforming to two different meta-model.

## 5. DISCUSSION

As discussed in section 2, we believe that model composition is a matter of 3 concerns: what, where and how to compose. The 3 selected approaches made different decisions concerning the separation of these concerns. Kompose proposes to let the user focus on the what only (by the signatures definitions) and hides the other two concerns in the tool. AMW allows a DSL engineer to define combination operators while designing a weaving metamodel. The users then specify what elements and how they want them to be composed by building the weaving model. On the other hand, MATA regroups all the 3 concerns in a single artifact, resulting in a compact representation of the model composition specification. Software development experience shows us that separating concerns leads to an easier development and maintenance, at the cost of having to manage several artifacts. Does this reasoning hold for model composition specification too? Should model composition concerns be separated, or kept together in a single compact document?

Section 4 discusses the flexibility offered by the three selected tools for composing models. Flexibility ranges from a fixed composition behavior to a full featured transformation language. However, this flexibility comes at a cost. The more flexible the approach is, the more the users have to provide additional configuration to the composition operator. Indeed, Kompose, the least flexible tool (as long as directives are not used) requires little parametrization. If the weaving metamodel and the ATL transformation are given, AMW uses only a model to drive the composition. On the opposite side, MATA asks the users to provide a complete transformation. What is the flexibility that is really needed

when composing model? What is the cost users are ready to pay for that flexibility? Is it possible to design a tool that can adapt its flexibility upon demand?

These are questions that should be addressed by researchers in the model composition community.

## 6. CONCLUSION

The main contribution of this paper is a framework for model composition that allows one to compare different composition approaches. The framework is given as a process. This conceptualization of model composition strongly contrasts with the idea of operator that has prevailed so far. Our process can be summarized as follows: decide what, where and how to compose. We support our claim by comparing three existing model composition techniques, for that purpose we make explicit the mechanisms they rely on to implement each step of our process. Furthermore, our proposed process provides several variation possibilities. While enumerating the variations the selected tools provide, we highlighted the cost of flexibility in terms of configuration that the users have to provide to the tools. Even though we do not provide answers to critical questions, we hope that these insights will help in designing the next generation of model composers.

## 7. REFERENCES

- [1] P. A. Bernstein, A. Y. Halevy, and R. A. Pottinger. A vision for management of complex models. *SIGMOD Rec.*, 29(4):55–63, 2000.
- [2] J. Bézivin, S. Bouzitouna, M. D. Del Fabro, M. P. Gervais, F. Jouault, D. S. Kolovos, I. Kurtev, and R. F. Paige. A canonical scheme for model composition. In A. Rensink and J. Warmer, editors, *ECMDA-FA*, volume 4066 of *Lecture Notes in Computer Science*, pages 346–360. Springer, 2006.
- [3] G. Brunet, M. Chechik, S. Easterbrook, S. Nejati, N. Niu, and M. Sabetzadeh. A manifesto for model merging. In *GaMMA '06: Proceedings of the 2006 international workshop on Global integrated model management*, pages 5–12, New York, NY, USA, 2006. ACM Press.

- [4] S. Clarke and E. Baniassad. *Aspect-Oriented Analysis and Design*. Addison-Wesley Professional, 2005.
- [5] F. Fleurey, B. Baudry, R. France, and S. Ghosh. A generic approach for automatic model composition. In *Aspect Oriented Modeling (AOM) Workshop*, Nashville, USA, October 2007.
- [6] R. France, I. Ray, G. Georg, and S. Ghosh. Aspect-oriented approach to early design modelling. *Software, IEE Proceedings -*, 151(4):173–185, 6 Aug. 2004.
- [7] C. Herrmann, H. Krahn, B. Rumpe, M. Schindler, and S. Völkel. An algebraic view on the semantics of model composition. In D. H. Akehurst, R. Vogel, and R. F. Paige, editors, *ECMDA-FA*, volume 4530 of *Lecture Notes in Computer Science*, pages 99–113. Springer, 2007.
- [8] I. Jacobson and P.-W. Ng. *Aspect-Oriented Software Development with Use Cases (Addison-Wesley Object Technology Series)*. Addison-Wesley Professional, 2004.
- [9] P. K. Jayaraman, J. Whittle, A. M. Elkhodary, and H. Gomaa. Model composition in product lines and feature interaction detection using critical pair analysis. In G. Engels, B. Opdyke, D. C. Schmidt, and F. Weil, editors, *MoDELS*, volume 4735 of *Lecture Notes in Computer Science*, pages 151–165. Springer, 2007.
- [10] 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, 2005.
- [11] Y. R. Reddy, S. Ghosh, R. B. France, G. Straw, J. M. Bieman, N. Mceachen, E. Song, and G. Georg. Directives for composing aspect-oriented design class models. *Trans. Aspect-Oriented Software Development*, pages 75–105, 2006.
- [12] J. Whittle, A. Moreira, J. Araújo, P. K. Jayaraman, A. Elkhodary, and R. Rabbi. An expressive aspect composition language for uml state diagrams. In G. Engels, B. Opdyke, D. C. Schmidt, and F. Weil, editors, *MoDELS*, volume 4735 of *Lecture Notes in Computer Science*, pages 514–528. Springer, 2007.