Model Integration Through Mega Operations

Th. Reiter, E. Kapsammer, W. Retschitzegger

Department of Information Systems
Johannes Kepler University Linz
{reiter | ek | werner}@ifs.uni-linz.ac.at

W. Schwinger

Department of Telecooperation
Johannes Kepler University Linz
wieland.schwinger@jku.ac.at

Abstract

With the advent of the Model Driven Architecture, models are replacing code as the major artifact in software development. A critical success factor for this is the possibility to derive models from each other in terms of transformations. Existing approaches, such as the forthcoming QVT-standard, will provide a proper foundation for transforming models on a fine-grained level. They, however, do not provide appropriate abstraction mechanisms for different integration scenarios, such as integrating models representing cross-cutting concerns or integrating models even from different domains. This paper proposes so called mega operations representing an abstraction mechanism which allow to specify model integration at the meta-level, thus forming the prerequisite to automatically derive a set of directives carrying out the actual integration at the model level. To cope with different integration scenarios, tight as well as loose integration of models is supported on top of a QVT-like language.

1. Introduction

The OMG has initiated the Model Driven Architecture (MDA), a software design methodology emphasizing the construction of models and the subsequent generation of executable code on basis of those models [23]. Thus models are replacing code as the major artifact in software development [4]. One of MDA's main benefit is the abstraction of core business functionality from implementation specific details resulting in platform independent models (PIM) and platform specific models (PSM).

The derivation of a model from another model is carried out through a *transformation* - ideally automated. In order to standardize such a model transformation language, several proposals for a Query/Views/Transformations (QVT)-language have been submitted to the OMG [12]. Model transformations as envisaged by QVT focus on transforming a model m_a at the level M1 conforming to a meta-model M_a at the level M2 into a model m_b conforming to a meta-model M_b, where M_a and M_b may potentially be the same.

QVT definitely represents a major building block technology for basic model transformations in the MDA. It does not provide, however, appropriate abstraction mechanisms for different kinds of *model integration scenarios*, which are highly needed in practice and well-known from other research areas such as *federated information systems* [29], [32], *megaprogramming* [31], *web service composition* and [19] and *aspect-oriented programming* [18]. Such integration scenarios would require a series of basic model transformations which will simply not scale up when manually specified for complex models.

Following, for example, the basic principle of separation of concerns in the modeling realm would avoid the construction of large, monolithic domain models which are difficult to handle and comprehend. At the same time, these models, each of them describing a certain cross-cutting concern of a whole domain (e.g., security aspects and transactional aspects of a web-based tourism information system), need to be *tightly integrated* into one coherent model representing the entire domain, as required for MDA.

Integration is not only needed in the case of models representing aspects of the same domain, but also in case of models covering different domains. For example, it would be highly desirable to integrate web-based reservation systems covering different domains like transportation (e.g, car rental) and accommodation (e.g., hotel booking) in order to allow them to interoperate providing new services for customers. This scenario requires for loose integration, i.e., synchronizing both domain models in certain ways by explicitly representing the model's interrelationships, while providing their autonomy.

Although these two scenarios look quite differently at a first sight, they bear several commonalties in mind. Both call for integrations which can be defined in an *abstract and thus, scalable way*, without burden the modeler with transformation primitives. Integration should not have to be defined repeatedly each time when models should be integrated, but rather be specified once at a meta-level, thus *facilitating reuse of integration knowledge*. Finally, in order to prevent ad-hoc integration of models, the *actual integration at the model level* should be *governed* at the meta level and performed fully *automated*.

To deal with these requirements and based on our experience with various web-based model integration scenarios (cf. [15], [16], [20], [26], [28]) we introduce so-called *mega operations*[1] providing abstraction mechanisms for model integration, thus allowing modelers to develop web systems of several interrelated models. Mega operations offer a set of operators for dealing with model heterogeneity as well as synchronization and provide the possibility to specify integration constraints. Specified at the level of meta-models that are MOF-based [24], a set of integration directives can be automatically derived, carrying out the actual integration at the model level.

To fulfill the needs of the integration scenarios outlined above, integration done by mega operations are required to follow two strategies. Facilitating the integration of aspect models into a coherent domain model, a so called *weaving mega operation* is proposed, achieving tight model integration. Integrating independent domain models requiring, e.g., synchronization and loose coupling to preserve their autonomy is supported by a so called *sewing mega operation*. These mega operations are based on a common architecture using primitive QVT-transformations underneath.

This paper introduces these two mega operations in Section 2 and 3, together with a set of appropriate operators for each of them. In Section 4, an architecture supporting these mega operations is outlined. After a detailed discussion of the benefits of our approach with respect to other closely related approaches in Section 5, we conclude the paper pointing out further research issues.

[1] The term "mega operation" is influenced by the notion of *megaprogramming* - a DARPA research program conducted in the late 1980's and early 1990's (cf. [5], [31]) - and *megamodel* which is defined to be a model, whose elements represent models [4].

2. Weaving

Weaving provides for a tight integration of models. This means that a model m_a conforming to a meta-model M_a and a model m_b conforming to a meta-model M_b, can be woven to produce a model m_{ab} which in turn conforms to a woven meta-model M_{ab} (cf. Fig. 1).

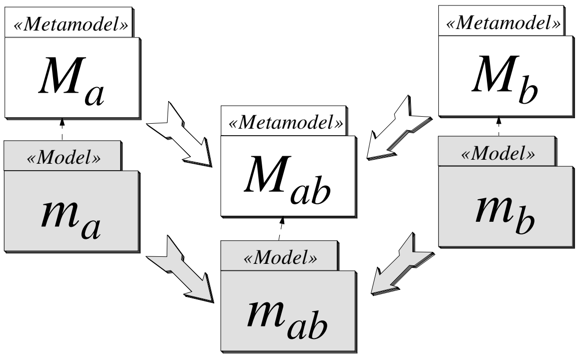

Figure 1. The Weaving Mega Operation

Note that the term weaving is adopted from aspect-oriented programming (AOP) [18], where it describes the process of weaving code representing a cross-cutting-concern into a base program. Transferring this basic idea into the modeling realm, but differently to the concept of weaving introduced by Bézivin et al. [3] (cf. Section 5), our weaving mega operation encompasses two steps:

(1) The weaving of *aspect meta-models* each of them describing a certain cross-cutting concern produces a *woven meta-model* (cf. Fig. 2)

(2) The subsequent weaving of *aspect models*, produces a *woven model* (cf. Fig. 3).

These two steps run automatically, provided that a particular *weaving specification* defines how to execute the weaving mega operation (cf. below).

Instead of only recording the semantic relationships between model elements, creating a woven model is necessary in case further processing or code generation mechanisms require so.

Furthermore, an advantage of defining a woven meta-model prior to the weaving of models is the ability to perform *conformance checks* with respect to the woven meta-model. Furthermore, having a meta-model for any given model is beneficial for *defining transformations* in the sense of MDA.

In the following we use a simplistic, though still sufficient running example to illustrate our basic ideas, stemming from the well-known domain of petri nets [25].

The meta-model M_{Petri} describes some basic structural aspects of a petri net consisting of places and transitions connected by arcs, whereas the meta-model M_{Mark} represents the aspect of markings, constituting places and marks (cf. Fig. 2).

The first step of the weaving mega operation - meta-model weaving - results in a woven meta-model $M_{PetriMark}$, representing a petri net with certain markings.

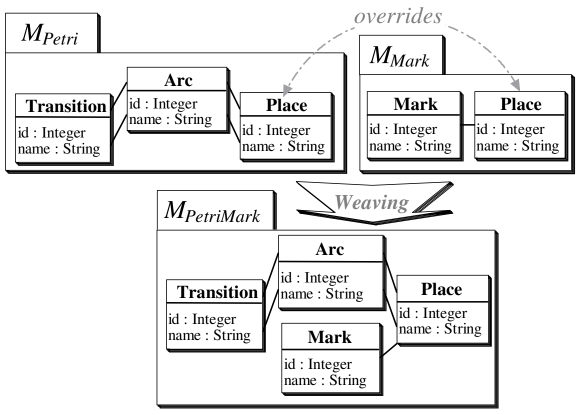

Figure 2. Meta-Model Weaving

The second step deals with the weaving of models (cf. Fig. 3). Governed by the new woven meta-model $M_{PetriMark}$, the woven model $m_{PetriMark}$ is produced, which consists of model elements from both source models m_{Petri} and m_{Mark}.

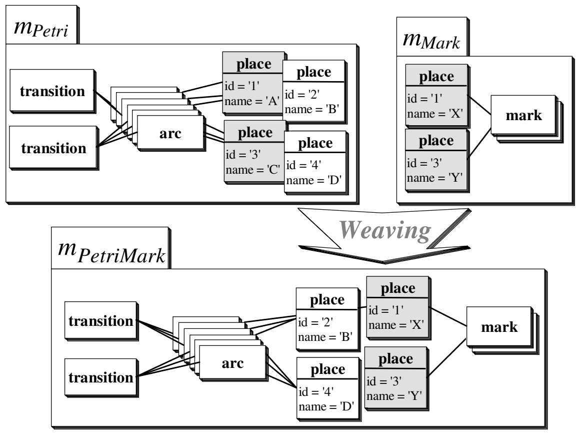

Figure 3. Model Weaving

Applying weaving mega operations as described above may yield the following benefits:

- Weaving allows the composition of domain meta-models, and thus enables the *re-use* of previously existing *domain knowledge*.
- Weaving allows several *teams* to model independently and weave their models as needed.
- Weaving allows the *evolution of a domain*, as new concerns can be woven in the form of aspect models, and thus supports incremental development of models.
- Weaving supports *scalability*, as there are no monolithic meta-models and models impairing comprehensibility.
- Weaving allows *libraries of models* to be built up for later re-use.
- Weaving makes modeling an activity of *assembling pre-existing "components"*.

The following subsections discuss the pre-requisites for putting the weaving mega operation into use, in terms of the weaving specification, comprising *weaving operators* and *model integration constraints*.

2.1. Weaving Operators

This subsection discusses several operators which are essential for defining weaving operations. Such operators need to address the reconciliation of overlapping concepts and allow basic model re-organization (cf., e.g., [30]). The set of operators comprises `overrides`, `references`, `prune`, and `rename` and does not claim to be complete.

Overrides. In case that two meta-models overlap in the form of elements representing the same concept, a weaving specification has to denote how to reconcile these model elements. Adopted from [30], but in contrast to them applied at a meta-level (cf. Section 5), we make use of an `overrides` operator which specifies that one meta-model element (qualified by "::") and its properties at the left hand side, take precedence over another meta-model element at the right-hand side.

With respect to the example shown in Fig. 3, the `overrides` operator expresses that meta-model element `Place` of the meta-model M_{Mark} replaces its pendant within the meta-model M_{Petri}.

M_{Mark}`::Place` **`overrides`** M_{Petri}`::Place;`

References and Inherits. If two meta-models do not conceptually overlap, a `references` operator denotes to connect meta-model elements via a new association. This operator also allows to specify the multiplicities of the association established between two meta-model elements. In our example, this expresses the fact that a place is able to hold an arbitrary number of marks.

M_{Mark}::Mark **references(*,1)** M_{Petri}::Place;

Similar to the references operator, but naturally not allowing for specifying multiplicities, we use an inherits operator to connect model elements via inheritance relationships.

Prune and Rename. As the previously introduced weaving operators "enrich" meta-models with elements, only, they cannot deal with the renaming or the deletion of possibly obsolete model elements, as portrayed in [30].

Therefore, a prune operator serves to rid all unnecessary meta-model elements in a meta-model. The example below shows the pruning of the obsolete Mark element.

M_{Mark}::Mark **prune**;

Renaming of meta-model elements can be done by applying a rename operator. As opposed to the previously mentioned weaving operators, prune and rename are unary in terms of meta-model elements. The example below shows the name change of the Place element.

M_{Petri}::Place **rename**('State');

2.2. Model Integration Constraints

Besides weaving operators, a weaving specification shall contain certain constraints, called *model integration constraints (MIC).* MICs are used to restrict the application of a weaving operator when integrating at the model level, thus forming some kind of precondition.

A MIC can be annotated for each application of a weaving operator. This means that the application of the operator at the model level is only carried out for those model elements, meeting the corresponding constraint. Thus, the MIC acts like a "filter", sorting out all invalid weaving operations and is indicated after the keyword "MIC:".

As shown in Fig. 3, only the Place model elements with id='1' and id='3' from the model m_{Mark} override the Place model elements in the m_{Petri} model with the matching values. For this, the previous example of the overrides operator is extended by the following MIC-specification:

MIC: M_{Mark}::Place.id **==** M_{Petri}::Place.id;
M_{Mark}::Place **overrides** M_{Petri}::Place;

2.3. Performing Meta-Model Weaving and Subsequent Model Weaving

A weaving operation can be reduced to a set of QVT transformations on the meta-model as well as on the model level. For the generation of the woven meta-model QVT transformations, derivable from the weaving specification, can be specified on the transformation's meta-level (M3) and applied to meta-models. In this way QVT populates the woven meta-model with model elements stemming from the meta-models to be woven. Likewise the subsequent weaving of models is specified in QVT on the transformation's meta-level (this time on M2) and applied to the model level. The actual QVT transformations to apply depend on the weaving operators involved and their attached MICs resulting in a certain transformation behavior.

According to the latest QVT 2.0 proposal [26] and to the best of our knowledge, Fig. 4 depicts an example transformation which could be derived from a weaving as shown in Fig. 2 and Fig. 3. Assuming that a transformation executed beforehand has populated the $m_{PetriMark}$ model with the model elements from m_{Petri}, the execution of the transformation below in the direction of the $m_{PetriMark}$ model, would enforce the overriding of place model elements and the creation of the according mark model elements.

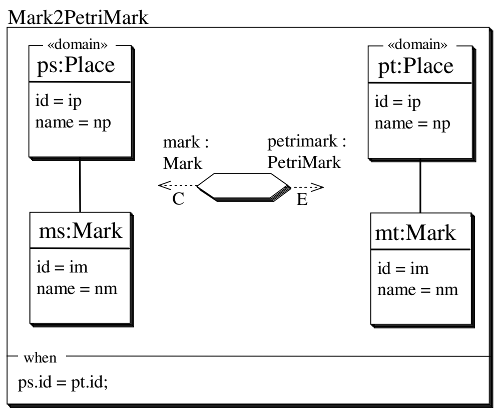

Figure 4. Example QVT Transformation

3. Sewing

As already mentioned, the weaving mega operation provides for a tight integration of models, by composing a coherent domain model from aspect models. Besides that, a loose coupling of models is required to integrate independent models pertaining to different domains and to keep them autonomous at the same time.

Therefore, apart from weavings, we see the necessity to introduce another mega operation called *sewing*. Sewing seems an appropriate analogy, as loose coupling can be seen as a form of stitching the involved models together, and thereby connecting without modifying them.

Analogous to weaving, a model m_a conforming to a meta-model M_a and a model m_b conforming to a meta-model M_b can be sewn to produce a set of *mediators* [32] realizing the integration, by "supervising" the sewn model elements (cf. Fig. 5).

Similar to a weaving specification, a *sewing specification* consists of operators annotated with MICs, and thus defines how to execute a sewing mega operation (cf. below). Specifying sewings on meta-models prior to the sewing of models, is deemed necessary to enable a meta-modeler to clearly define which model elements are valid to be sewn, and to henceforth rule out the ad-hoc creation of possibly ill-defined sewings.

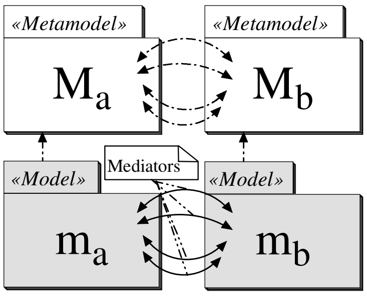

Figure 5. The Sewing Mega Operation

Continuing our running petri net example, let's imagine that we would like to have a graphical user interface (GUI) for a petri net simulation (cf. Fig. 6). The mega operator sewing could establish (similar to the model-view-controller paradigm) a loose coupling between the `name` attribute of the `Place` meta-model element belonging to the petri net model, and the `title` attribute of a `TextField` meta-model element belonging to the GUI model.

A tight coupling in the form of weaving the GUI model and the petri net model would not be adequate in this situation, as different domains are involved and weaving would simply entangle the different domain concepts.

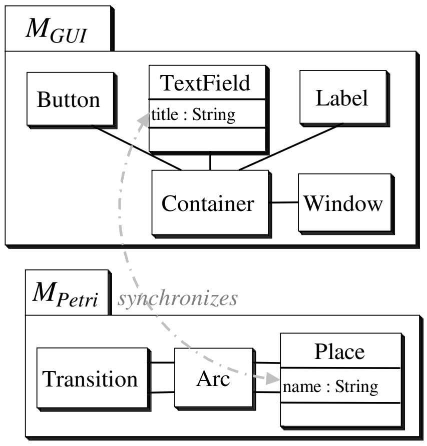

Figure 6. Meta-Model Sewing

As shown in Fig. 7, the application of a sewing mega operation at the meta-level results in the establishment of mediators between model elements, guided by MICs.

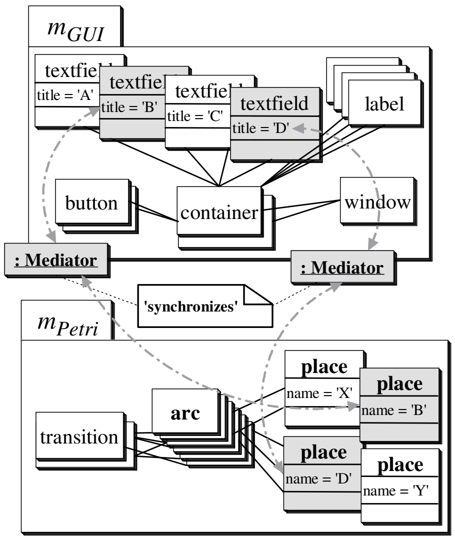

Figure 7. Model Sewing

Applying sewing mega operations as described above may yield the following benefits:

- Sewing integrates models, but still allows them to exist independently without affecting their structure and thus keeping their *autonomy*.
- Sewing serves to keep models *synchronized*.

- Sewing integrates models pertaining to different domains *without entangling* their concepts.

The following subsections discuss sewing operators together with their corresponding MICs and realization in terms of mediators.

3.1. Sewing Operators

The particular behavior of mediators depends on the specific operators in the sewing specification. The following subsection introduces such operators, namely `synchronizes` and `depends`, which are useful for the sewing of models. Such operators enforce to supervise the sewn model elements by observing their states and appropriately propagate changes.

Synchronizes. In case that, for instance, attributes of two model elements should be kept synchronized, a `synchronizes` operator can be used to denote that fact. With respect to the previous example (cf. Fig. 6 and Fig. 7) the `synchronizes` operator together with a MIC is employed as follows:

```
MIC:   MGui::TextField.title ==
       MPetri::Place.name;
MGui::TextField.title synchronizes
MPetri::Place.name;
```

According to the MIC, synchronizations are established between `TextField` model elements and `Place` model elements, only, if having equal values for their `title` and `name` attributes, respectively. Applied on the model level (cf. Fig. 7), changing the value of the `title` attribute would lead to a change in the value of the `name` attribute.

Depends. The `depends` operator is used to denote that the existence of one model element depends on the existence of another. If two teams are working on two separated, though related models, it can be useful to establish such correspondences between the related model elements. Thus, if one team decides to delete a model element, the related model element should immediately be deleted as to avoid inconsistencies among the teams' models. The example below shows a sewing specification for the meta-model element `TextField` depending on the meta-model element `Place`.

```
MIC:   MGui::TextField.title ==
       MPetri::Place.name;
MGui:TextField depends MPetri::Place;
```

It has to be noted, that sewing is focused on integrating existing models, not on creating them anew from another model, as QVT allows. Sewings therefore have a narrower domain and aim at simplifying certain integration tasks that would probably be more cumbersome to express using QVT alone.

3.2. Sewing realized by Mediators

The application of a sewing operator does not result in a newly produced, integrated model per se, as it is the case with weaving, where heterogeneities in the form of conceptual overlap can be eliminated through the establishment of woven meta-models and models. On the contrary, sewing has to handle, or better to say, transparently resolve existing overlap throughout sewn models. Thus, the outputs of the sewing mega-operation are mediating entities producing the desired integration behavior.

On the model level, mediators can manifest as QVT transformations propagating attribute changes or creating and deleting model elements accordingly.

Operators other than the two previously introduced `depends` and `synchronizes`, which would for instance allow model elements to be transparently connected via associations and generalizations across model boundaries, could be realized using the *Java Metadata Interface* (JMI) [10] and the *Eclipse Modeling Framework* (EMF) [9]. They provide an infrastructure for the generation of programming interfaces to instantiate and manipulate models as Java run-time objects. Such programs resulting from sewn models have to be adapted in a way, as to reflect the semantics and the mediating behavior of the specific operator. In case that it is not possible to influence the model code generation, an elegant solution would be to utilize an aspect-oriented approach and weave the necessary code fragments for the mediator pattern into the model code. The aspect code necessary would be derived from the sewing specifications.

However, when finally code is to be produced from models, the mediating behavior also has to be realized on the system level, specific to a certain platform. Sewings can of course manifest as models themselves, which describe the respective semantics and the integration behavior imposed on models. The generation of platform specific "bridge" code facilitating a loose coupling on the system level is thus rendered a common task like any other model driven development, as integration of heterogeneities is taken care of on the model level. The mediation on the system level could for instance be carried out by a web service, connected to different systems generated from sewn models.

4. Architecture

This section proposes a first sketch of an architecture for the implementation of a mega-operation toolkit and briefly discusses relevant technologies. Fig. 8 shows a GUI component as means for handling a Mega-Operations Controller, which orchestrates the toolkit's components as required. A MOF repository serves as basis for storing meta-models and models. To access and manipulate them programmatically, programming interfaces like JMI or EMF Java mappings can be used. Although EMF and JMI provide the necessary infrastructure for manipulating models, they are not capable of model transformations in the sense of QVT. Hence, a QVT-like model transformation tool such as Marius[2], which has been developed in the course of a former cooperation between the University of Linz and the University of South Australia, is employed for model transformation. Enforcing constraints on models can be accomplished by an Object Constraint Language (OCL) checker like [1].

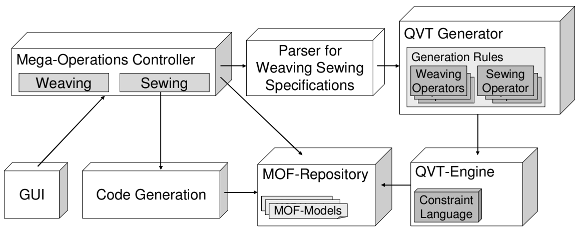

Figure 8. Architecture for Mega Operations

As already mentioned, the weaving mega operation can essentially be expressed as a series of QVT-like transformations, as can the sewing mega-operation concerning the model level. Thus, weaving and sewing specifications are parsed and input into a QVT generator, which can be seen as the toolkits core component, "compiling" weavings and sewings into QVT-code. The resulting code is in turn executed by a QVT-engine upon models stored in the repository to achieve the integration of models. A code generation component serves to create bridge code realizing the sewing mega-operation's loose coupling on the system level. The therefore necessary "glue" code can be incorporated into the code derived from models either directly through customisation of the generated code or through an aspect weaver like AspectJ [18].

[2] The name *Marius* stems from Gaius Marius, a Roman consul and general, best known for initiating a series of reforms 107 B.C., completely restructuring the organization equipment and tactics of the Roman army.

5. Related Work

This section gives an overview on other approaches most relevant with respect to our idea of mega operations. For this, the main focus of each approach is summarized briefly, followed by clearly pointing out similarities and differences to our own approach.

Table 1 summarizes the results by giving an overview on operators supported as well as whether the approach deals with arbitrary MOF-based models either on the M1 or the M2 level and if the specification on M2 is used for model integration on M1.

Table 1. Comparison of Related Approaches

							Weaving Operators				***Sewing*** Operators	
		MOF-based	M2 meta-level	M1 model-level	Meta-level based integration	MICs	overrides	reference	inherits	prunes / renames	synchronizes	depends
Approach	AMMA	✓	✓	✓	✓	▨	▨	▨	▨	▨	✓	▨
	Rondo	▨	✓	✓	▨	▨	✓	▨	▨	✓	▨	▨
	Model Composition Semantics	▨	▨	✓	▨	▨	✓	✓	▨	▨	▨	▨
	Model Composition Directives	▨	▨	✓	▨	▨	✓	✓	▨	✓	▨	▨
	GME	▨	▨	✓	~	▨	✓	✓	✓	▨	▨	▨
	C-SAW	▨	▨	✓	✓	▨	▨	▨	▨	▨	▨	▨
	Domain Composition Approach	▨	▨	~	▨	▨	▨	▨	▨	▨	✓	▨

Legend:		
✓	...	explicitly supported
▨	...	not explicitly supported
~	...	not applicable

AMMA. Bézivin et al. [3], [4], [21] are developing the *Atlas Model Weaver (AMW)* as part of the *AMMA model engineering platform*, which is soon to be released under the *Eclipse GMT project* [8]. The AMW aims at supporting modelers to establish semantic links between elements of different models or meta-models, which can serve as input for further tools. Model weaving in the sense of Bézivin et al. seems to be a manual operation specifying links between elements of different models or meta-models. The set of links produced by such a weaving operation is represented by a *weaving model*. A weaving model appears to be similar to a *weaving specification* in our approach, which specifies operators linking meta-model elements.

Our approach, however, extends the notion of weaving from an activity that establishes semantic links between meta-models, to a mechanism that actually interprets operators specified between meta-model elements and carries out operations accordingly. These operations involve the automatic generation of a new woven metamodel, which is an integration of the original metamodels. Furthermore, we provide a mechanism to automatically integrate models into a new woven model conforming to the new woven meta-model. In our understanding, weaving is treated as a distinct abstraction mechanism for the integration of both, models *and* meta-models.

Rondo. Within the *Generic Model Management* initiative, Bernstein et al., [2], [22] work on merging meta-data in the form of relational schemata and XML schemata. *Rondo* is an implementation thereof, providing model management operators that enable modelers to deal with models rather than model elements. Similar to our weaving and sewing operators, these operators include a *match* operator, which automatically establishes semantic correspondences between similar schema elements and a *merge* operator allowing to combine different model elements.

In contrast to them, however, we explicitly focus on MOF-models in the sense of MDA, keeping a later code-generation step following model integration in mind. Furthermore, in our approach, a meta-modeler is able to specify the integration of models and meta-models on a meta-level, instead of providing generic model management operators to manipulate models.

Model Composition Semantics. Clark [7] introduces a *composition mechanism* for UML class diagrams. This approach deals with the composition of models representing different separated concerns. Overlapping concepts are identified in these models and thus merged as specified by a composition relationship, following so-called *merge* and *override* strategies. Merge integration for example applies when equivalent classes appear in multiple design models, and conflicts need to be reconciled among these. Override integration can be used to substitute obsolete parts of a design with new modeling constructs. Based on these basic integration behaviours, *composition patterns* [6] are introduced as an extension to UML templates.

This approach, however, focuses on UML models, only, and does not provide for deletion of obsolete model elements after a weaving is performed, as required for our approach.

Model Composition Directives. Based on [7], Straw et al. [30] propose so called *composition directives* for composing UML class diagrams. These basically include name rewriting, adding and deleting of model elements, change of references, and control of execution order. Inspired by aspect-oriented programming concepts, so-called primary models are composed with aspect models, which represent a cross-cutting-concern to be interwoven.

Although composition directives are comparable to our weaving operators, their primary focus seems to be on model weaving but not on meta-model weaving. We believe that our mega operations could in turn be transformed into composition directives at the model level. Since we avoid an ad-hoc integration of models, with our mega operations, licit integrated models can be generated, only.

GME. The *Generic Modeling Environment (GME)* proposed by Karsai et al. [17] is a modeling and meta-modeling toolkit based on UML notation and a GME-specific meta-metamodel. GME allows for the composition of meta-models similar to our approach. The composition mechanisms comprise an *equivalence operator* creating a union of two model elements, similar to the *merge* semantics in [7] and two different inheritance operators, realizing implementation inheritance and interface inheritance.

One major difference to our approach is that GME is not based on the MOF standard. Furthermore, we believe that our approach goes beyond the functionalities for meta-model composition in the GME by introducing model integration constraints, allowing even fine-grained integration of models.

C-SAW. C-SAW, developed as a plug-in for the above-mentioned GME by Gray et al. [13], [14], is a so-called *cross-cutting-concern weaver*. Aspects are specified using the *Embedded Constraint Language (ECL)*, which is a superset of OCL, additionally providing imperative constructs for model manipulation.

The transformation capabilities of ECL are, however, limited to models of the same meta-model and it lacks support for abstract integration mechanisms as supported by our approach.

Domain Composition Approach. Estublier et al. [9] propose a *UML profile* to allow the composition of separately designed domain models, as required when facing the federation of immutable components off the shelf. UML associations and association classes are specialized by dedicated stereotypes to express feature correspondence and concept overlapping.

In principle, this approach is similar to our sewing mega operation. In contrary to this UML-based approach, our sewing mega operation is applicable to arbitrary MOF models. In addition, it seems that their focus lies not on tight integration of models, as done by our approach.

6. Conclusion and Outlook

This paper proposes mega-operations for model integration and shows the benefits that can be gained thereof. Apart from QVT-like mappings, which can be seen as the base requirement to the MDA approach, the introduced mega-operations weaving and sewing provide abstraction mechanisms to cope with complex modeling scenarios, allowing for a tight and loose coupling, respectively. Thus, enhanced scalability and further re-use capabilities of a model-driven approach are gained.

Future work will especially concentrate on clearly defining the integration behavior enforced by weaving and sewing operators.

Therefore, on the one hand the proposed operators have to be specified in detail, and on the other hand, further operators have to be conceived. Detailing would, e.g., include clarifying different reconciliation behaviors of the `overrides` operator, propagation behavior of the `synchronizes` operator, as well as detecting and resolving conflicts arising from the application of the mega-operations.

With respect to both, weaving and sewing, a clear syntax and means for representing the mega-operations as MOF models have to be developed.

Furthermore, an important issue to resolve will be to find ways to derive platform specific implementations for mediators.

Finally, a prototypical implementation for mega-operations shall be developed. Experiments with this prototype should yield valuable insight into the applicability of mega-operations as devised in this paper.

References

[1] D. Akehurst, O. Patrascoiu, “OCL 2.0-Implementing the Standard for Multiple Metamodels”, *Proc.s of the UML'03 workshop*, Electronic Notes in Theoretical Computer Science, November 2003.

[2] P. A. Bernstein, “Applying Model Management to Classical Meta Data Problems” *Proc. of the Conf. on Innovative Database Research (CDIR03),* Asilomar, California, Jan. 2003, pp. 209-220.

[3] J. Bézivin, F. Jouault, P. Valduriez, “First Experiments with a ModelWeaver”, *OOPSLA & GPCE Workshop*, Vancouver, October 2004.

[4] J. Bézivin., F. Jouault, P. Valduriez, “On the Need for Megamodels”, *OOPSLA & GPCE Workshop*, Vancouver, October 2004.

[5] B. Boehm, B. Scherlis, “Megaprogramming”, *Proceedings of the DARPA Software Technology, Conference,* 1992.

[6] Clarke, S., Walker, R.J. “Composition Patterns: An Approach to Designing Reusable Aspects”, *Proceedings of International Conference on Software Engineering (ICSE)*, Toronto, Canada, 2001.

[7] S. Clarke. “Extending standard UML with model composition semantics”, *Science of Computer Programming, Elsevier Science*, Volume 44, Issue 1, July 2002, pp. 71-100.

[8] Eclipse Foundation, *Generative Model Transformer (GMT),* http://www.eclipse.org/gmt/, 2005.

[9] Eclipse Foundation, *Eclipse Modeling Framework (EMF)*, http://www.eclipse.org/emf, 2005

[10] Java Community Process, *Java Metadata Interfaces (JMI)*, 2002, http://java.sun.com/products/jmi/

[11] J. Estublier, A. D. Ionita, G. Vega, “A Domain Composition Approach”, *Proc. of the International Workshop on Applications of UML/MDA to Software Systems (UMSS),* LasVegas, USA, June 2005.

[12] T. Gardner, C. Griffin, J. Koehler, R. Hauser, “A review of OMG MOF 2.0 Query / Views / Transformations Submissions and Recommendations towards the final Standard”, *Object Management Group (OMG)*, ad/2003-08-02.

[13] J. Gray, T. Bapty, S. Neema, A. Gokhale, “Generating Aspect-Code from Models”, *OOPSLA Workshop on Generative Techniques for Model-Driven Architecture*, Seattle, WA, November 2002.

[14] J. Gray, T. Bapty, S. Neema, D. C. Schmidt, A. Gokhale, B. Natarajan, “An Approach for Supporting Aspect-Oriented Domain Modeling”, *Generative Programming and Component Engineering (GPCE)*, Springer-Verlag LNCS 2830, Erfurt, Germany, September, 2003, pp. 151-168.

[15] G. Kappel, E. Kapsammer, W. Retschitzegger “Integrating XML and Relational Database Systems”, *World Wide Web Journal (WWWJ), Kluwer Academic Publishers*, Vol. 7(4), December 2004, pp. 343-384

[16] E. Kapsammer, W. Schwinger, W. Retschitzegger, “Bridging Relational Databases to Context-Aware Services”, *Proc. Of the CAiSE Workshop on Ubiquitous Mobile Information and Collaboration Systems (UMICS)*, Springer LNCS, Porto, Portugal, June 2005.

[17] G. Karsai, M. Maroti, A. Ledeczi, J. Gray, and J. Sztipanovits, “Composition and Cloning in Modeling and Meta-Modeling Languages”, *IEEE Transactions on Control System Technology*, special issue on Computer Automated Multi-Paradigm Modeling, March 2004, pp. 263-278.

[18] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, L. Videira, J.-M. Loingtier, J. Irwin, "Aspect-Oriented Programming", *Proc. Of the European Conference on Object-Oriented Programming (ECOOP), Springer LNCS 1241*, Finland, 1997.

[19] J. Koehler and B. Srivastava, „Web service composition: Current solutions and open problems.", Proc. of the *ICAPS, Workshop on Planning for Web Services*, Trento, Italy, June 2003.

[20] G. Kramler, E. Kapsammer, G. Kappel, W. Retschitzegger, "Towards Using UML 2 for Modelling Web Service Collaboration Protocols", *Proc. of the First Int. Conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA)*, Geneva, Switzerland, February 2005.

[21] D. Lopes, S. Hammoudi, J. Bézivin, F. Jouault, "Mapping Specification in MDA: From Theory to Practice", *First International Conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA)*, Geneva, Switzerland, February 2005.

[22] S. Melnik, "Generic Model Management: Concepts and Algorithms", *Springer LNCS 2967*, 2004.

[23] Object Management Group, "MDA Guide", Version 1.0.1, June 2003 [http://www.omg.org/docs/omg/03-06-01.pdf]

[24] Object Management Group (OMG), "MOF 2.0 IDL Specification", July 2004, [http://www.omg.org/cgi-bin/apps/doc?ptc/04-07-01.pdf]

[25] C. A, Petri, "Fundamentals of a Theory of Asynchronous Information Flow", *Proc. of IFIP Congress 62, Amsterdam: North Holland Publ. Comp.,* 1963, pp. 386-390.

[26] QVT-Merge Group, "Revised Submission for MOF 2.0 Query/View/Transformation RFP(ad/2002-04-10)", *Version 2.0, ad/2005-03-02,* March 2005

[27] Th. Reiter, "Transformation of Web Service Specification Languages into UML Activity Diagrams", *Master Thesis, Dept. of Information Systems, Johannes Kepler University Linz,* March 2005.

[28] M. Schrefl, M. Bernauer, E. Kapsammer, B. Pröll, W. Retschitzegger, T. Thalhammer, "Self-Maintaining Web Pages", *Information Systems (IS), International Journal*, Vol. 28/8, Elsevier Science Ltd., 2003, pp. 1005-1036

[29] A.P. Shet, J.A. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases", *ACM Computing Surveys*, Vol. 22, No 3., Sep. 1990, pp. 182-236.

[30] G. Straw, G. Georg, E. Song, S. Ghosh, R. France, and J. M. Bieman, "Model Composition Directives", *7th UML Conference*, Lisbon, Portugal, October, 2004.

[31] G. Wiederhold, P. Wegner, S. Ceri. "Toward Megaprogramming", *Communications of the ACM*, November 1992.

[32] G. Wiederhold, "Mediators in the Architecture of Future Information Systems", *IEEE Computers*, Vol. 25, No. 3, March 1992, pp. 38-49.