Electronic Communications of the EASST
Volume 42 (2011)

Proceedings of the 4th International Workshop on Multi-Paradigm Modeling (MPM 2010)

On the Unification of Megamodels

Regina Hebig and Andreas Seibel and Holger Giese

12 pages

Guest Editors: Vasco Amaral, Hans Vangheluwe, Cécile Hardebolle, Lazlo Lengyel
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

On the Unification of Megamodels

Regina Hebig[1] **and Andreas Seibel**[2] **and Holger Giese**[3]

[1]regina.hebig@hpi.uni-potsdam.de [2]andreas.seibel@hpi.uni-potsdam.de
[3]holger.giese@hpi.uni-potsdam.de
Hasso Plattner Institute at the University of Potsdam
Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany

Abstract: Through the more and more widespread application of model-driven engineering (MDE) and the increasing diversity in applied modeling paradigms within single projects, there is an increasing need to capture not only models in isolation but also their relations. This paper is a survey on techniques capturing such relations, such as megamodels or macromodels, based on existing scientific literature. Therefore, we consider various definitions of these techniques. We further examine characteristics of the different techniques. We will propose a unified core definition of a megamodel that captures the core properties of megamodels and which can be extended to the needs of the different applications of megamodels. Finally, we give an outlook on arising application areas for megamodels.

Keywords: megamodel, multi-modeling, model management, model-driven engineering

1 Introduction

Today, model-driven engineering (MDE) is used in an increasing number of software development projects. In 1997, OMG proposed the first version of the unified modeling language (UML), which is a general purpose modeling language for software development.

Thereby, an increasing amount of models of different modeling languages are applied to develop and realize the desired system. These models may represent different complementing aspects of the system under development, e.g., user interface, application logic, etc. Furthermore, models may also show overlapping aspects of the system under development but at different levels of detail, which is suggested in the unified process [JBR99] by developing a system in different phases (e.g., requirements, analysis, design, implementation, etc.). Additionally, a key feature of MDE is automation. Thus, models are automatically created from other models by means of model transformations.

However, a big challenge in MDE-based development projects is the fact that models do not exist in isolation and changes occur frequently, which render existing models to be inconsistent. Dependencies between models are manifold, e.g., overlapping aspects in the system under development, application of model transformations, etc. If inconsistencies are not resolved or stay undetected, it can be a thread to the success of MDE-based projects. Thus, the ability to explicitly deal with dependencies between models can strongly influence the success of an MDE-based development project.

Consequently, in recent years scientific approaches in the area of multi-paradigm modeling arose, which deal with the question of how to capture the dependencies between models in

different modeling languages. Prominent examples are *megamodels* proposed by J. Bézivin [BGMR03] and J.M. Favre [Fav04] or R. Salay's macromodels [Sal07][1]. The dependencies of models that already exist are not the only thing in focus of research. Approaches that deal with transformation chains [DABJ08, FBV+09] aim to predefine and automatically perform compositions of model transformations that are defined between models.

There is already a considerable amount of literature about megamodels and related approaches [BGMR03, Fav04, Sal07, SME09, BJV04, BJRV05, ABBJ06, BFB07, Fav05, FN05, PBR09, SNG09].[2] However, the understanding and desired usage of megamodels differs from author to author. Therefore, we collect characteristics that can reasonably be applied to megamodels for classifying existing approaches and for providing an overall understanding of megamodels. In order to provide a common understanding of the concept megamodel, we unify the different definitions and provide a corresponding metamodel that reflects the definition. With the help of the characteristics we span a space of actual application scenarios for megamodels. To the best of our knowledge there is currently no approach that gives an overview about megamodel approaches or provides a unified definition.

In Section 2 we survey different megamodel-related approaches. In Section 3 we present a unified definition for the term megamodel. We summarize possible application areas of megamodels in Section 4 and conclude in Section 5.

2 State of the Art

In this section, we discuss approaches that define the term 'megamodel' or related concepts like the 'macromodel'. We focus on how terms are defined and on the intention for the application of these types of models. Thereby, we highlight differences between the definitions as characteristics of the approaches. Talking about an approach, we refer to a set of papers that are published by a shared set of authors and clearly seem to describe the same view of the term megamodel or related concept, respectively. In order to reference an approach throughout the paper, we have selected one author from each such group of papers as 'representative' author.

2.1 Definition of Megamodels and Related Terminologies

Most approaches that deal with megamodels come up with their own definition of the term. In the following we introduce the different definitions.

BÉZIVIN is one of the first founders of the term megamodel. His view of the term megamodel has grown during the recent years. The first definition was given in the year 2003: *"A megamodel is a model with other models as elements"* [BGMR03]. In 2007 this definition is further complemented: *"A megamodel contains relationships between models"* [BFB07].

FAVRE can be seen also as one of the founders of the term megamodel. Nevertheless, in comparison to BÉZIVIN he has a strong theoretical focus on megamodels. In 2004 he gave the following definition: *"... the idea behind a megamodel is to define the set of entities and relations that are necessary to model some aspect about MDE"* [Fav04]. In a follow-up paper, FAVRE provided an extension of the definition: *"A megamodel is a model that*

[1] We will use the term megamodel for all approaches throughout this paper, since the main part of the surveyed papers use this term.

[2] Due to space limitations we focus on five approaches and omitted approaches that strongly overlap with them.

represents this kind of complex arrangements without entering into the details of each artifact" [Fav05].

SALAY provides an approach that uses the term macromodel, which is related to the term megamodel but has a different intention. A first definition is provided in 2007: *"A macromodel is a graphical model whose elements denote models and whose edges denote model relations"* [Sal07]. Later, he defines the term macromodel as: *"A macromodel consists of elements denoting models and links denoting intended relationships between these models with their internal details abstracted away"* [SME09].

PEROVICH provides another definition of the term megamodel in [PBR09]: *"A megamodel is a model composed of related models"*.

SEIBEL is also influenced by the terminology of BÉZIVIN and provides the following definition: *"A megamodel is a combination of high-level traceability models and low-level traceability models"* [SNG09]. A high-level traceability model shows thereby relations between models whereas a low-level traceability model shows relations between element of different models.

2.2 Comparing the definitions

All approaches have in common that they use megamodels or macromodels in order to illustrate the dependencies between models. We further want to compare the definitions used by the different approaches. Thereby, some definitions come along with a metamodel for megamodels. These are the definitions of SEIBEL, FAVRE, BÉZIVIN, and SALAY. BÉZIVIN, FAVRE, and SALAY further allow extensions of their metamodels, such that the possible relations can be adapted to the current needs ('extensibility').

Besides the opportunity to extend the metamodel, some approaches have constraints on possible megamodels. For example, the metamodel that is introduced by BÉZIVIN excludes that a megamodel can be used as reference model. There, a megamodel is modeled as terminal model. Other constraints can arise if the types of relations that are allowed to be part of the megamodel are predefined. Most definitions do not restrict the possible relations, although they predefine relation types. BÉZIVIN and PEROVICH only predefine the 'conformsTo' relation for associating a model to its reference model. However, he does not restrict the possible types of relations between the models. SALAY does not restrict the types of relations within the macromodel. In SEIBEL's approach relations are named 'traceability links', which can be fact links or obligation links. While fact links only state that there is a relationship, obligation links also define how a change can be propagated to restore the consistency of a relationship. Furthermore, SEIBEL allows the definition of the type of the traceability links via the 'isOfType'-relation. Thus, SEIBEL does not restrict the possible relation types, since the type and semantic of a required traceability link can be defined by the user of the approach. FAVRE introduces his metamodel in order to identify a set of relations that are sufficient to cover the needs of MDE. Naturally, the possible relation types are restricted to the relations already identified by FAVRE, such as 'RepresentationOf' , 'DecomposedIn', 'ConformsTo', or 'IsTransformedIn'. New relation types would require an extension of this metamodel. Therefore, we say that FAVRE restricts the possible types of relations.

	SALAY	FAVRE	BÉZIVIN	PEROVICH	SEIBEL
Provide Metamodel	✓	✓	✓	-	✓
Extensibility	✓	✓	✓	-	-
Constraints (Megamodel is no Reference Model)	-	-	✓	-	-
Constraints on Relations	-	✓	-	-	-
Hierarchy of Megamodel	✓	✓	✓	✓	✓
Hierarchy of relations	✓	-	-	-	✓
Inter-Level-Relations	✓	-	-	-	✓

Table 1: Properties of different Megamodel-definitions

As a further characteristic of a megamodel definition we want to highlight the question whether it supports *hierarchy*. That includes on the one hand the question, whether a megamodel can contain further megamodels. On the other hand there is the question whether relations can contain other relations. The most approaches allow megamodels explicitly to contain further megamodels. FAVRE allows systems to contain further systems via the 'DecomposedIn' relation. Similarly, SEIBEL allows a 'subElement' relation between models. Since a model that contains models conforms to the definition of a megamodel, one might say that SEIBEL allows a megamodel to contain further megamodels. BÉZIVIN allows a megamodel to contain models, while he sees a megamodel as a model. Thus, BÉZIVIN allows a megamodel to contain further megamodels. In a similar manner SALAY allows the same. Finally, PEROVICH provides an explicit use case for a megamodel that contains another megamodel. He uses megamodels that contain models, which show different related views of the same system, to illustrate the state of a system at a specific point in development (e.g., the computation independent architecture). In an outer megamodel PEROVICH shows the dependencies between different states of a system under development.

SALAY and SEIBEL allow relations to contain further relations. BÉZIVIN does not directly allow a relation to contain a further relation, but provides weaving models that can be associated to relations and contain relations between the elements of models that are associated to that relation. There is a further characteristic that is fulfilled by SEIBEL. He allow '*inter-level-relations*', this means relations that connect modeling artifacts on different hierarchical levels. For example, such a relation can connect a model element with a model. Following the formal definition of macromodels in [Sal07] SALAY also allows '*inter-level-relations*'.[3]

2.3 Application of Megamodels

After looking at the definition of the term megamodel we want to summarize the intended applications of the different approaches. Thereby, we add the approach of DIDONET as example for transformation chain approaches. However, since these approaches deal with a kind of model operation, one can consider to use megamodels as definition language for transformation chains.

BÉZIVIN proposes different applications of megamodels. In [BGMR03, BJV04, BJRV05, ABBJ06] megamodels are applied to support model-driven software development by using it for model management. Thus, megamodels provide a global view on models. Whereas,

[3] Confusingly, SALAY's metamodel in [SME09] seems to allow no relations between macromodels and models.

in [BFB07] megamodels are applied to facilitate traceability between models and their elements. In [BJB08] it is shown how to apply megamodels to the management of complex systems like the Bugzilla system in Eclipse.

FAVRE applies megamodels to model MDE [Fav04, Fav05, FN05]. He does not focus on the applicability of megamodels but on reasoning about relations that can exist in the context of MDE. He defines exemplary patterns in MDE by using the megamodels.

SALAY shows in [Sal07, SME09] how to apply macromodels to capture the modelers intention how different models, showing different views of the system, are related to each other. Therefore, a macromodel can be a type that represents a pattern that defines a certain intention of a modeler and it can be an instance that represents the current models. He provides an automated analysis to examine whether the intention of the modeler is satisfied.

PEROVICH facilitates megamodels to design software architectures in [PBR09]. In his approach a megamodel defines a software architecture and design decisions are encoded in the megamodel as model transformations that are related to relations between models.

SEIBEL shows in [SNG09], similar to [BFB07], that megamodels are predestined for the application of traceability in MDE. Additionally, relations can be associated with behavior, such as model transformations. This 'behavior of relations' is used to resolve inconsistencies that may occur when models change.

M. DIDONET introduces in [DABJ08] an approach for supporting the combination of rules and automated execution of multiple transformations. Thereby, he already proposes the integration of their language into a megamodeling platform, such that the interrelation of multiple transformation, models and metamodels can be better handled.

2.4 Characteristics of Megamodel Approaches

Besides the intention of a megamodel, it is interesting to look at the operations that are performed on a megamodel. SEIBEL, BÉZIVIN, and PEROVICH automatically *adapt* their megamodels to the current situation. In contrast, SALAY *compares* his macromodels with the current situation and verifies, whether the constraints defined in the macromodel are satisfied. In addition to the automated derivation of the megamodel, SEIBEL automatically *interprets* his megamodels in order to restore consistency between models. Also FAVRE interprets his megamodels in order to identify mega-pattern automatically. Since DIDONET does not use megamodels, he cannot perform operations on it. However, the specification he uses instead is interpreted within the tool.

	SALAY	FAVRE	BÉZIVIN	PEROVICH	SEIBEL	DIDONET
Compare megamodel	✓	-	-	-	-	-
Adapt megamodel	-	-	✓	✓	✓	-
Interprete megamodel	-	✓	-	-	✓	(✓)

Model of Behavior	-	-	-	-	✓	✓
Model of Structure	✓	✓	✓	✓	✓	-
Overlap-Relation	✓	-	✓([BFB07])	✓	✓	-
Model Operation-Relation	-	✓	✓([BGMR03])	✓	✓	(✓)
Static-Relation	✓	✓	✓	✓	✓	-
Descriptive	✓	✓	✓	✓	✓	-
Prescriptive	✓	-	-	-	✓	✓

Table 2: Characteristics of megamodels approaches

Models in classical software engineering are used to illustrate the structure, the behavior, and the function of a system (e.g., feature models [SHT06]). Transferred to megamodels '*behavior*' means that the approach treats the megamodel as a behavioral model and '*structure*' means that the approach does only consider the megamodel as a representation of the models including their relationships. Most approaches use megamodel for the illustration of structure, which might, for example, be compared to a current situation. Only SEIBEL also defines behavior, since he defines the trigger for actions to reestablish consistency as well as the corresponding actions within the megamodel. None of the approaches uses megamodels to illustrate function.

We differentiate between four types of relations. Thereby, the first type is the relation that illustrates a relation that exists in the represented system. Typically this type of relation occurs in 'normal' models and is in most cases not expected to occur in megamodels. The other three types of relations should not be illustrated with non-megamodel models. First, there is the *'overlap'-relation* that indicates that models have overlapping aspects. This relation is, for example, used by SALAY and PEROVICH to indicate that two models show different views of the same system. SEIBEL uses this relation as basis for identifying consistency and inconsistency between two models. In [BFB07] BÉZIVIN uses megamodels for traceability issues and not as repository as in [BGMR03]. Therefore, he uses 'overlap'-relations in [BFB07]. The *'model operation'-relation* is a representative of a model operation that was/can be/will be performed with models as input and models as output. For example, FAVRE introduces the 'isTransformedIn' relation, which indicates that one model was transformed to another model. Similarly, BÉZIVIN and PEROVICH support this relation type. SEIBEL uses the 'model operation'-relation for automatically restoring the consistency between two models. The last type of relation is called *'static'-relation*. We introduce this type for the reason of capturing a huge amount of relations, such as 'conformsTo', which represents the relation between a model and a reference model, or the 'contains' relations, which indicates that a model is contained by another model. This might also include the 'represents' relation between a models an the represented system, which is used by FAVRE. We do not provide a full list of relations that can be handled as 'static'-relations. Like FAVRE denotes, there is still research on the question of which relations between models are part of MDE. Most approaches, like BÉZIVIN and PEROVICH, work with the 'conformsTo' relation. SEIBEL uses the 'isOfType' relation for links. Also SALAY uses to show the 'instanceOf' relation in his example models. A model might be used to describe a current situation (*descriptive*) or to define a desired situation or behavior (*prescriptive*). The approaches of BÉZIVIN and PEROVICH consider the megamodel to be created automatically and, thus, to describe the actual state of the system.

Also SEIBEL works with an automatically derived megamodel and thus descriptive megamodels. However, SEIBEL defines operations for reestablishing consistency between models and, thus, prescribes also behavior. FAVRE captures a current picture, where he also models mega-patterns of relations, which he uses to describes possible types of activities in MDE. SALAY uses his megamodel to describe a current situation as well as to prescribe desired situations with constraints. As the goal of DIDONET is the definition of a chain of transformations, he specifies behavior, which is the application of transformations in the specified order. Thus, DIDONET's approach is prescriptive. Since, DIDONET uses no megamodel he defines no relations. However, he specifies transformations and, thus, deals with model operations.

3 Unified Definition of Megamodel

As shown above, megamodel-related approaches have different motivations, and come along with different meanings of the term megamodel. We first provide a 'unified' definition of the term megamodel, which is a synthesis of the definitions of the investigated approaches from Section 2.1. Based on this unified definition we provide a generic metamodel of a megamodel. Finally, we define what we expect from a megamodel.

The basic structure of a model can be considered as *a graph that contains nodes and edges between them*. A classical software model is typically a graph [Küh05] and thus is *a model that contains model elements and relations between them* with model elements are nodes and relations are edges. A megamodel could be considered as a classical software model. Thus, we define a megamodel as: *a model that contains models and relations between them*. However, there is a major difference to classical software models that is a megamodel explicitly considers models instead of model elements. This difference is visually shown in Figure 1.

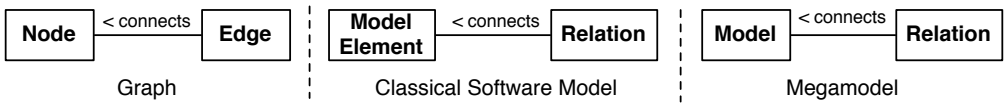

Figure 1: Core of structural models

Based on this unified definition of a megamodel we define a metamodel, which is free of implementation aspects and can be extended to the needs of the different approaches. The metamodel is shown in Figure 2. This metamodel contains additional concepts like models can contain models, models can contain relations and relations can depend on other relations. These additional concepts are motivated in the following.

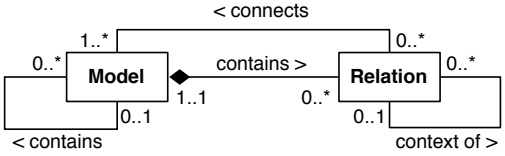

Figure 2: Core metamodel for megamodels

But first, why is a relation not considered to be a model? Basically, because an edge is not considered as a graph. In the megamodel case, the relation itself only describes that models are related in some specific way (dependent on the type of a relation). For example, FAVRE introduces a relation of the type 'isTransformedIn', which reflects that one model is the result of a transformation that was applied to another model. If an approach like SEIBEL's requires for

example a transformation specification to be associated to a relation, the core metamodel can be extended, such that a relation can have a corresponding association to a model.

A model can contain models for at least two reasons. First, a megamodel is a model that contains models and thus a model should be capable of containing models (hierarchical megamodel). In this case, the model is a megamodel that can contain models as well as megamodels. Second, model elements in classical software models can be interpreted as models and thus models can contain models. In this case, the model is a model element, which does not contain further models. However, this depends on how models are internally structured and how they are interpreted. To the best of our knowledge, there is currently no consensus about how to structure models right. A prominent example is UML. Technically, a UML model is a single monolithic model. However, conceptually a UML model may contain different models, e.g, class diagrams, sequence diagrams, etc. Nevertheless, these are only model elements in a UML model. This interpretation of model elements depends on the level of detail that should be considered in a megamodel.

Additionally, a model can contain relations. A megamodel is a model that contains relations between models. Thus, the model should reflect the capability of containing relations, too. In the core metamodel relations are not directed. A direction can be introduced by extending the core metamodel. Finally, a relation between models does not necessarily exist in isolation. Certain relations may require another relation. In [FBV+09] a transformation explicitly requires another transformation as its own context. In [SNG09] it is shown that relations at different levels of detail can have a dependency between each other. We further expect that models in a megamodel, which are not megamodels itself, do only contain relations that are necessary to describe their original. All other relations should not be part these models but should be explicitly contained in the megamodel that contains these models. For example, relations like 'conformsTo', 'contains' (if they are used for structuring models only), 'isTransformedIn', etc. should be explicitly reflected in a megamodel but not in the models itself.

All in all, our proposed core metamodel of megamodels is extensible, allows hierarchies of models (and thus of megamodels) and relations, and supports 'inter-level-relations'. In comparison to BÉZIVIN, we do not provide specific types of models nor relations. Furthermore, we do not declare a megamodel to be a terminal model, which is a specific kind of model. FAVRE's metamodel of megamodels does not distinguish between the concept of models and relations. He also predefines specific relations (e.g., 'conformsTo') in the metamodel, which is however not necessary in all application domains of megamodels. SALAY's metamodel of macromodels does explicitly distinguish between macromodel and model and further between macrorelations and relations. We do not make this explicit distinction because we define a megamodel to be a model. SEIBEL's metamodel of megamodels is somehow similar to this metamodel but it is more detailed. It explicitly distinguished between models and model elements. Furthermore, it is distinguished between two kinds of relations. Subsuming, our proposed core metamodel provides a consistent view on megamodels, which covers all surveyed approaches.

To give an idea of how to extend the core metamodel of the megamodel, we show an exemplary extension in Figure 3. (a) shows a metamodel extension, which additionally contains three kinds of relations like 'Transformation', 'conformsTo' and 'isTransformedIn' and four kinds of models like 'M2Model', 'M1Model', 'M1ModelElement' and 'Tool'. 'M2Model' represents metamodels whereas 'M1Model' represents models, which conform to 'M2Model' reflected by

the ‘conformsTo’ relation. The ‘Transformation’ relation is specified by a ‘M1Model’, which contains an executable specification of a model transformation. This specification can be executed by a tool ‘Tool’. It takes a transformation with a specification connected and creates an ‘isTransformedIn’ relation between M1Models it has transformed.

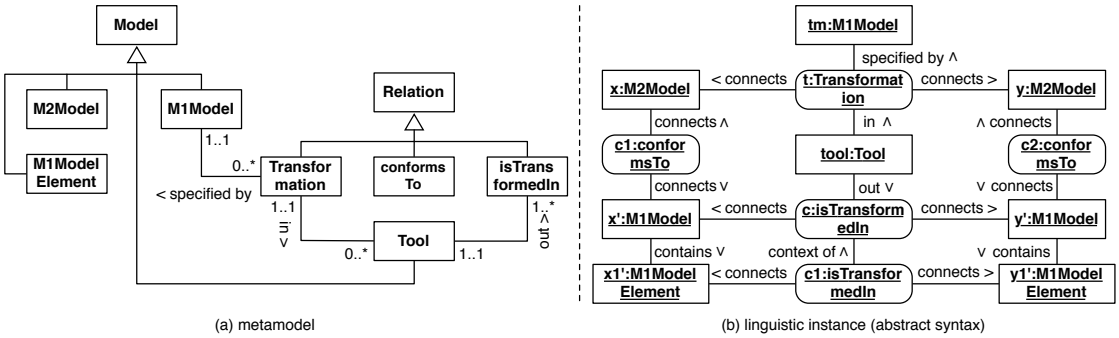

Figure 3: Exemplary extension of the core megamodel metamodel

(b) shows a linguistic instance[4] of the metamodel extension. ‘tm’ is the specification of a model transformation ‘t’ between the two ‘M2Model’-metamodels ‘x’ and ‘y’. ‘tool’ applied ‘t’ on ‘x” (which conforms to ‘x’) and ‘y” (which conforms to ‘y’)[5]. Thereby, ‘tool’ produced the ‘isTransformedTo’ relation ‘c’. A strength of the hierarchy can be seen in the example, too. It is possible to illustrate not only that the model ‘x” was transformed to the model ‘y”, but also that in this context, the model element ‘x1” was transformed to the model element ‘y1”.

Note, this is only one possible extension of the megamodel-metamodel. For example, the semantic of the specified ‘isTransformedIn’ relation might not be appropriate for each megamodel approach. In (b) there is an ‘isTransformedIn’ relation between model ‘x” and model ‘y”. What happens if model ‘x” is affected by changes? In the semantic of this metamodel extension, there is no reason that the ‘isTransformedIn’ relation will be removed. This is appropriate if the changes on ‘x” do not influence ‘y′’ or if the megamodel approach is only interested in the fact, that the transformation happened. However, other megamodel approaches might require the transformation to be redone. It might even be desired to introduce the notion of validity to the transformation ‘isTransformedIn’, such that the relation would become invalid if ‘x” changes.

This example shows that even terms that are strongly associated to megamodels, such as transformations, can have quite different semantics in different megamodel approaches. This is one reason why we present such a minimalist metamodel. The second reason is introduced below.

4 Application perspective

The term megamodel, as it is introduced in the section above, is nearly reduced to its minimum. We chose this minimalist form to give the reader the space to think not only about what megamodels are, but also what megamodels can be. We state that megamodel can be applied to many different areas, such as system design, analysis, simulation, observation at runtime and model management. In this section we give a perspective on possible applications of megamodels.

An application area for structural megamodels is already indicated by SALAY, who defines

[4] We took the terminology ‘linguistic’ and ‘ontological’ instance from [Küh06].
[5] The ‘conformsTo’ relation reflects an ontological instance.

constraints on a megamodel. Structural megamodels can be used as reference models. Thereby, a reference megamodel might on the one hand describe how to compose a structural megamodel out of different model types, such that a desired system is sufficiently described by the different views provided by the models in the megamodel. For example, it might be desired that a design for a distributed system contains interaction diagrams for each interface defined in class diagram within the design. On the other hand, a structural reference megamodel can directly describe a system type, since it can set reference models for different views of a system in relation to each other. Corresponding to reference models a structural megamodel might also be used to specify 'anti pattern' for a desired system.

On structural megamodels different analyses might be performed. For example, a structural megamodel can be use for impact analysis or the evaluation of metrics, such as the number of interdependencies per model. Further a structural megamodel can be subject of adaption. This might be synchronizing the megamodel with changes in the illustrated models (bottom up). Alternatively, there is the application area of model management operations, where changes in the megamodel are propagated to the illustrated models (top down), as for example the creation or deletion of models. Furthermore, megamodels can be applied in different application domains, e.g., capturing models for modeling a system, capturing models for modeling modeling of a system (metamodeling), capturing runtime models, etc., or even a combination of those domains.

As shown in SEIBEL's approach, megamodels can be considered as behavioral models, which can be used to maintain consistency between models. Another application area is the definition of transformation chains, as it was already proposed by Didonet [DABJ08] and Fritzsche [FBV+09]. Such as structural megamodels also behavioral megamodels can be used as reference models. In contrast to the definition of a transformation chain, which can be automatically performed, a behavioral reference megamodel has a higher degree of freedom. For example, the behavioral reference megamodel might define that a certain set of model operations can be repeated, without specifying the exact number of repetitions. Further, a behavioral reference megamodel can include model operations that cannot be performed fully automatic, but have to be supported by the user.

Finally, we state that the big challenge concerning megamodels is the ability to deal with the complexity of hierarchies of nested models. Model-analysis and modeling techniques, such as transformations that are until now only designed for the 'classical' software models, will have to be developed further to be applied to megamodels with deep hierarchies. For example, a transformation might not be defined for a class diagram, but for a whole UML model, which is handled as megamodel. Furthermore, a megamodel can combine models in different modeling languages, for example combinations of different DSLs. Modeling techniques have to be extended to deal with that form of heterogeneity.

5 Conclusions & Future Work

Different approaches provided different understandings of the term megamodel. We introduced a unified definition of the term megamodel including a core metamodel that will help to improve discussions about megamodels. Furthermore, most approaches assume their understanding of 'relations between models' as obvious. We were able to show that the semantics of the relations in the presented approaches are different. With the help of the introduced characteristics, the megamodel approaches can be better understood. Many approaches combine different types

of relations. For example, although BÉZIVIN concentrates on 'model operation'-relations he allows 'conformsTo'-relations. PEROVICH goes a step further and even combines 'overlap'-relations with 'model operation'-relations and 'conformsTo'-relations. With this combination PEROVICH pays attention to the complexity of software development, where a system is not build through transforming a single model, but a whole set of models. SEIBEL combines 'overlap'-relations and 'model-operation'-relations in another manner. His approach combines both types in obligation links. Such links show overlaps between models, but define also operations for the reestablishment of consistency with respect to the overlaps. With this combination SEIBEL also combines structure and behavior in the same megamodel. Part of the contribution of this paper is that we can now identify such combinations.

Finally, we came up with a short outlook for research and future applications of megamodels. In future, we want to have a closer look to the types of possible relations in megamodel. We are currently working on a technical report, as a more detailed version of this paper.

Bibliography

[ABBJ06] F. Allilaire, J. Bézivin, H. Bruneliére, F. Jouault. Global Model Management In Eclipse GMT/AM3. In *Proc. of Eclipse Technology eXchange workshop (eTX) at ECOOP'06*. 2006.

[BFB07] M. Barbero, M. D. D. Fabro, J. Bézivin. Traceability and Provenance Issues in Global Model Management. In *Proc. of 3rd Workshop on Traceability (ECMDA-TW'07)*. Pp. 47–55. Haifa, Israel, June 2007.

[BGMR03] J. Bézivin, S. Gérard, P.-A. Muller, L. Rioux. MDA components: Challenges and Opportunities. In *Proc. of First International Workshop on Metamodelling for MDA*. Pp. 23 – 41. York, UK, November 2003.

[BJB08] M. Barbero, F. Jouault, J. Bézivin. Model Driven Management of Complex Systems: Implementing the Macroscope's Vision. In *ECBS '08: Proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems*. Pp. 277–286. IEEE Computer Society, Washington, DC, USA, 2008.

[BJRV05] J. Bézivin, F. Jouault, P. Rosenthal, P. Valduriez. Modeling in the Large and Modeling in the Small. In *Proc. of European MDA Workshops: Foundations and Applications (MDAFA'05)*. LNCS 3599/2005, pp. 33–46. Springer, 2005.

[BJV04] J. Bézivin, F. Jouault, P. Valduriez. On the Need for Megamodels. In *Proc. of OOPSLA/GPCE Workshop on Best Practices for Model-Driven Software Development*. Nashville, TN, USA, 2004.

[DABJ08] M. Didonet Del Fabro, P. Albert, J. Bézivin, F. Jouault. Industrial-strength Rule Interoperability using Model Driven Engineering. Research report RR-6747, INRIA, 2008.

[Fav04] J.-M. Favre. Foundations of Model (Driven) (Reverse) Engineering – Episode I: Story of The Fidus Papyrus and the Solarus. In *Proc. of Dagstuhl Seminar on Model Driven Reverse Engineering*. 2004.

[Fav05] J.-M. Favre. Megamodelling and Etymology. In *Proc. of Dagstuhl Seminar on Transformation Techniques in Software Engineering*. Volume 05161. 2005.

[FBV+09] M. Fritzsche, H. Brunelière, B. Vanhooff, Y. Berbers, F. Jouault, W. Gilani. Applying Megamodelling to Model Driven Performance Engineering. In *Proc. of 16th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS '09)*. Pp. 244–253. IEEE Computer Society, Washington, DC, USA, 2009.

[FN05] J.-M. Favre, T. Nguyen. Towards a Megamodel to Model Software Evolution Through Transformations. *Electr. Notes Theor. Comput. Sci.* 127(3):59–74, 2005.

[JBR99] I. Jacobson, G. Booch, J. Rumbaugh. *Unified Software Development Process*. Addison-Wesley, 1999.

[Küh05] T. Kühne. What is a Model? In *In proc. of Dagstuhl Seminar*. Volume 04101. 2005.

[Küh06] T. Kühne. Matters of (Meta-)Modeling. *Software and System Modeling* 5(4):369–385, 2006.

[PBR09] D. Perovich, M. C. Bastarrica, C. Rojas. Model-Driven approach to Software Architecture design. In *Proc. of ICSE Workshop on Sharing and Reusing Architectural Knowledge (SHARK '09)*. IEEE Computer Society, Washington, DC, USA, 2009.

[Sal07] R. Salay. Towards a Formal Framework for Multimodeling in Software Engineering. In *Proc. of the Doctoral Symposium at the ACM/IEEE 10th International Conference on Model-Driven Engineering Languages and Systems*. Volume Vol-26. Nashville (TN), USA, October 2007.

[SHT06] P.-Y. Schobbens, P. Heymans, J.-C. Trigaux. Feature Diagrams: A Survey and a Formal Semantics. In *Proc. of 14th IEEE International Requirements Engineering Conference (RE'06)*. Volume 0, pp. 139–148. IEEE Computer Society, Los Alamitos, CA, USA, 2006.

[SME09] R. Salay, J. Mylopoulos, S. Easterbrook. Using Macromodels to Manage Collections of Related Models. In *Proc. of 21st International Conference on Advanced Information Systems Engineering (CAiSE'09)*. LNCS 5565/2009, pp. 141–155. Springer, Amsterdam, The Netherlands, 8-12 June 2009.

[SNG09] A. Seibel, S. Neumann, H. Giese. Dynamic Hierarchical Mega Models: Comprehensive Traceability and its Efficient Maintenance. *Software and System Modeling* 9(4):493–528, 2009.