Towards a Formal Approach to Metamodel Evolution

Florian Mantz, Adrian Rutle, Yngve Lamo
Bergen Universiy College
P.O. Box 7030, 5020 Bergen, Norway
{fma,aru,yla}@hib.no

Alessandro Rossini, Uwe Wolter
University of Bergen
P.O. Box 7803, 5020 Bergen, Norway
{rossini,wolter}@ii.uib.no

1 Introduction

Model-driven engineering (MDE) is a trend in software engineering which aims at improving productivity, quality, and cost-effectiveness of software. This is obtained by considering models as first-class entities of the software development process and adopting model transformation to automate the implementation.

The word "model" has different meanings in different contexts. In software engineering, model denotes "an abstraction of a (real or language-based) system allowing predictions or inferences to be made" [4]. Models in software engineering are typically diagrammatic. The word "diagram" has also different meanings in different contexts. In software engineering, diagram denotes a structure which is based on graphs; i.e. a collection of nodes together with a collection of arrows between nodes. Since graph-based structures can be visualised in a natural way, "visual" and "diagrammatic" modelling are often treated as synonyms. In this work, visualisation and diagrammatic syntax are clearly distinguished. That is, the proposed approach focuses on precise syntax and semantics of diagrammatic models independent of their visualisation.

In the context of MDE, models are typically specified by means of modelling languages. Each modelling language has a corresponding metamodel. Models which are specified by a modelling language should conform to the metamodel of the language. Models as well as metamodels undergo a complex evolution during their life cycles. As a consequence, when a metamodel is modified, models conforming to this metamodel should be migrated in such a way that they conform to the modified version (see Fig. 1). This problem is referred to as metamodel evolution in the literature [2].

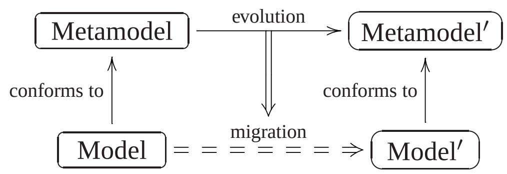

Figure 1: Metamodel evolution and model migration

Model migration can be achieved by means of model transformations, which are usually described by a set of transformation rules. In this sense, automatic model migration implies automatic definition of transformation rules. Unfortunately, this is not always feasible [2]. An interesting challenge is then to identify the metamodel modifications which allow for automatic derivation of transformation rules. Moreover, another challenge is to provide a precise characterisation of the conditions under which these transformation rules generate a model which conforms to the modified version of the metamodel. This work presents some ideas towards a formal approach to metamodel evolution which addresses these challenges. The proposed approach is based on the Diagram Predicate Framework (DPF) [7, 5, 8, 6] which provides a formalisation of (meta)modelling and model transformation based on graph theory [3] and category theory [1].

$$
\begin{tikzcd}
\mathfrak{L} \arrow[rr, "r"] \arrow[dr, "m"] & & \mathfrak{R} \arrow[dr, "m'"] & \\
& \mathfrak{S}_2 \arrow[rr, "r'"] & & \mathfrak{S}'_2 \\
I^{\mathfrak{L}} \arrow[uu, "\iota^L"] \arrow[rr, "r^*"] \arrow[dr, "m^*"'] & & I^{\mathfrak{R}} \arrow[uu, "{[\![r]\!](\iota^L)}"'] \arrow[dr, dotted, "m'^*"] & \\
& \mathfrak{S}_1 \arrow[uu, "\iota^{S_1}"] \arrow[rr, dotted, "r'^*"'] & & \mathfrak{S}'_1 \arrow[uu, dotted, "\iota^{S'_1}"']
\end{tikzcd}
$$

Figure 2: Rule-based model migration

In DPF, models are represented by *(diagrammatic) specifications*. A specification $\mathfrak{S} = (S, C^{\mathfrak{S}} : \Sigma)$ consists of an underlying graph S together with a set of *atomic constraints* $C^{\mathfrak{S}}$ [7, 6]. The graph represents the structure of the model while atomic constraints add restrictions to this structure. Atomic constraints are formulated by predicates from *(diagrammatic predicate) signatures*. A signature $\Sigma = (P^{\Sigma}, \alpha^{\Sigma})$ consists of a collection of predicates, each having a name, a shape graph, a visualisation and a semantic interpretation [7, 6]. For a given specification $\mathfrak{S}$, the semantics is given by the set of instances (I, ι) where $\iota : I \to S$ is a graph homomorphism which satisfies the constraints in $C^{\mathfrak{S}}$. We use $\mathsf{Inst}(\mathfrak{S})$ to denote the category of instances of $\mathfrak{S}$.

In view of DPF, metamodel modifications can be described by a set of constraint-aware transformation rule applications [8] (along with the matches and the sequence of the rules applications). These rules are sound with respect to the modifications; that is, applying the rules via the matches in the given sequence will lead to the same modified version of the metamodel. Moreover, the analysis of the metamodel evolution problem is based on categorical constructions such as pushout and pullback.

Fig. 2 outlines the relation between models, metamodels and model transformations in the context of metamodel evolution. $\mathfrak{S}_2$ and $\mathfrak{S}'_2$ represent two versions of a metamodel. $\mathfrak{L}$ and $\mathfrak{R}$ represent the left hand side and right hand side of a model transformation rule r which describe the metamodel evolution. $\mathfrak{S}_1$ represents a model which conforms to the metamodel $\mathfrak{S}_2$. This model should be migrated to $\mathfrak{S}'_1$ by a transformation rule r^* corresponding to transformation rule r on the metamodel level. Hence, the left hand side and right hand side of the transformation rule r^* have to conform to the left hand side and right hand side of r, respectively. After a closer look at Fig. 2 one may observe the following[1]:

- The top face is a pushout since r is a rule application.
- The left face is a pullback since an instance (S_1, ι^{S_1}) of $\mathfrak{S}_2$ can be restricted to an instance $(I^{\mathfrak{L}}, \iota^L)$ of $\mathfrak{L}$.
- The back face is a pullback too since the left hand side and right hand side of the model migration rule r^* can be restricted to the left hand side and right hand side of the model evolution rule r.

Furthermore, this construction requires additional properties to guarantee a conformance preserving model migration:

[1] The cube in Fig. 2 may be related to van Kampen Square [9]

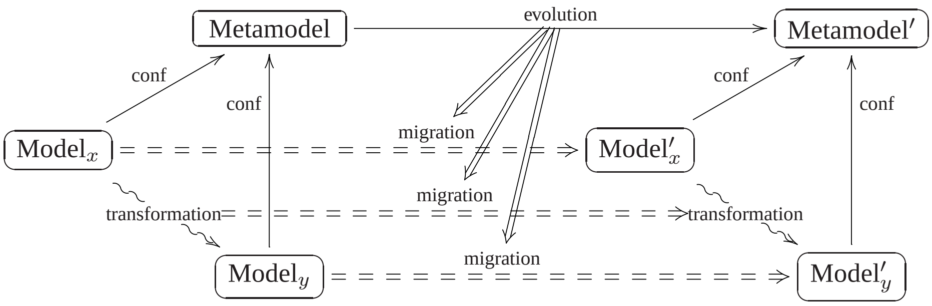

Figure 3: Functorial properties of model migration

– For a rule $r : \mathfrak{L} \to \mathfrak{R}$ the semantics $[\![r]\!] : \mathsf{Inst}(\mathfrak{L}) \to \mathsf{Inst}(\mathfrak{R})$ can be any mapping satisfying the condition that for all instances $(I^{\mathfrak{L}}, \iota^L) \in \mathsf{Inst}(\mathfrak{L})$ the back face is a pullback.

– If the bottom face is a pushout, then the front face and the right face have to be pullbacks.

Another interesting research line in this topic is the ability to also obtain higher order transformation rules from metamodel modifications. Higher order transformation rules have as their input and output other transformation rules. In this way, the technique outlined above may be used to transform both models and relations between them. Hence, it would be appropriate to consider under which conditions model migration has functorial properties (see Fig. 3).

References

[1] M. Barr and C. Wells. *Category Theory for Computing Science (2nd Edition)*. Prentice Hall, 1995.

[2] A. Cicchetti, D. Di Ruscio, R. Eramo, and A. Pierantonio. Automating Co-evolution in Model-Driven Engineering. In *EDOC 2008*, pages 222–231. IEEE Computer Society, 2008.

[3] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. Springer, March 2006.

[4] T. Kühne. Matters of (meta-)modeling. *SoSyM*, 5(4):369–385, 2006.

[5] A. Rossini, A. Rutle, Y. Lamo, and U. Wolter. A Formalisation of the Copy-Modify-Merge Approach to Version Control in MDE. *JLAP*, 79(7):636–658, 2010.

[6] A. Rutle. *Diagram Predicate Framework: A Formal Approach to MDE*. PhD thesis, Department of Informatics, University of Bergen, Norway, 2010.

[7] A. Rutle, A. Rossini, Y. Lamo, and U. Wolter. A Diagrammatic Formalisation of MOF-Based Modelling Languages. In *TOOLS 2009*, volume 33 of *LNBIP*, pages 37–56. Springer, 2009.

[8] A. Rutle, A. Rossini, Y. Lamo, and U. Wolter. A Formalisation of Constraint-Aware Model Transformations. In *FASE 2010*, volume 6013 of *LNCS*, pages 13–28. Springer, 2010.

[9] U. Wolter and Z. Diskin. From Indexed to Fibred Semantics – The Generalized Sketch File. Technical Report 361, Department of Informatics, University of Bergen, Norway, October 2007.