Composition of Multilevel Modelling Hierarchies

Alejandro Rodríguez[1], Adrian Rutle[1], Francisco Durán[2], Lars Michael Kristensen[1], Fernando Macías[3], and Uwe Wolter[4]

[1] Western Norway University of Applied Sciences, Bergen, Norway
[2] Universidad de Málaga, Málaga, Spain
[3] Universidad de Extremadura, Cáceres, Spain
[4] University of Bergen, Bergen, Norway

Introduction. Model-driven software engineering (MDSE) has been proven to be a successful approach in terms of a gain in quality and effectiveness [15]. It tackles the constantly growing complexity of software by utilizing abstractions and modelling techniques.

MDSE promotes separation of concerns to better handle the complexity of software systems, which in turn leads to the creation of several models that need to be composed when reasoning about the overall system. Traditional MDSE approaches such as the Eclipse Modelling Framework (EMF) [14] and the Unified Modelling Language (UML) [1] are based on the two-level modelling approach: one for (meta)models and one for their instances. This enforces model designers to specify systems within only two abstraction levels, which in turn may lead to challenges like model convolution and accidental complexity [5]. Multilevel modelling (MLM) addresses these challenges by eliminating the restriction in the number of times a model element can be instantiated. Indeed, MLM has proven to be a successful approach in areas such as software architecture and process modelling domains [5, 2]. In this context, MLM techniques match well with the creation of domain-specific modelling languages (DSMLs), especially when we focus on behavioural languages since behaviour is usually defined at the metamodel level while it is executed at least two levels below; i.e., at the instance level.

Our approach for MLM. It is based on the idea that one must be able to specify models which are both generic and precise [8]. This encompasses not only the definition of structure but also behaviour. We specify behaviour descriptions by defining in-place model transformations (MTs) which are rule-based modifications of an initial model that give rise to a transition system. We have proposed in previous work the so-called Multilevel Coupled Model Transformations (MCMTs) as a means to overcome the issues of both the traditional two-level transformation rules and the multilevel model transformations. While the former lacks the ability to capture generalities, the later is too loose to be precise enough (case distinctions) [9]. We use our own tool MultEcore [7] to specify both the structure and the semantics of multilevel hierarchies and rely on our infrastructure which utilizes Maude as an execution engine [12]. The Maude specification automatically created by MultEcore can be used for simulation and analysis [11]. It is also possible to further conduct reachability analysis and model checking. While the former can be done by means of strategies [4], the latter can be performed through the model checker that Maude implements.

Composition of MLM DSMLs. One of the most successful techniques in MDSE is the definition of DSMLs. Even though they aimed to specific domains, many of them share certain commonalities coming from similar modelling patterns [10]. Needless to say, composition is key in achieving interoperability among these DSMLs. In this paper, we focus on the theoretical constructions for composition of multilevel DSMLs by presenting two approaches (depicted in Fig 1). Our framework is founded on graph transformations and category theory. The composition of modelling hierarchies would be carried out by pushout construction in the category of graph chains (see [6]), while the composition of the transformation rules (MCMTs) would be

carried out by the amalgamation of these rules. We plan to formalise the latter by extending our formalisation through an adaptation of the constructions in [3] where the amalgamation of two-level DSMLs is formally described.

Several approaches pursue composition of languages by defining a merge operator. Intuitively, "the common elements are included only once, and the other ones are preserved". Fig.

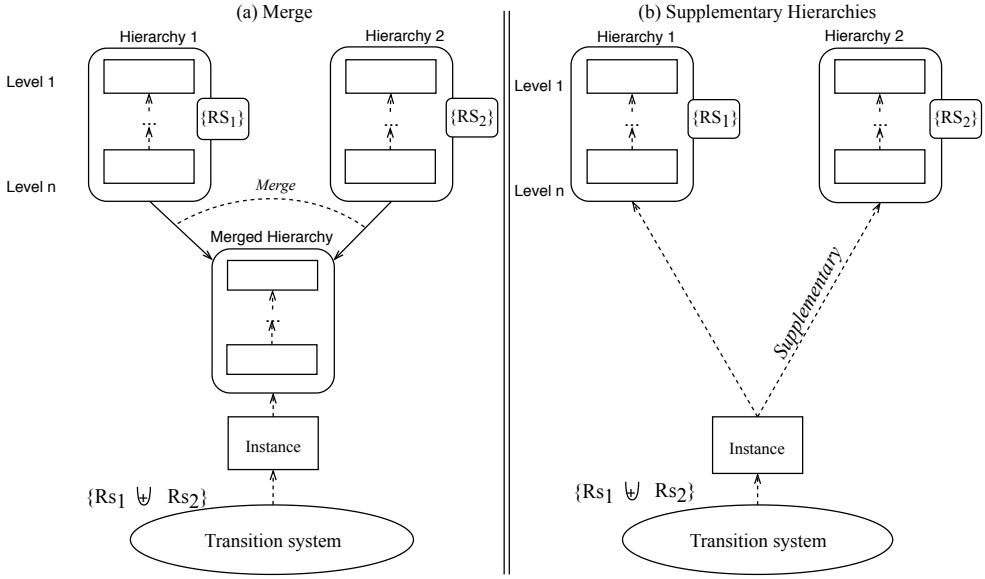

Figure 1: Conventional merge vs supplementary referencing composition

1a depicts how merge would be defined over two multilevel hierarchies, each one representing a DSML. The Merged Hierarchy is the result of merging the involved multilevel hierarchies. Instance models can now instantiate elements (dashed arrows represent typing graph homomorphisms) that come from the merging process. These instances can be executed producing the Transition system which is obtained by applying the rules that come from the amalgamation of the rule sets (RS) of each hierarchy ($RS_1 \uplus RS_2$).

Our approach for composition. In our approach, the modeller typically works with a multilevel hierarchy which we identify as the *application hierarchy*. Application hierarchies can optionally include an arbitrary number of *supplementary hierarchies* which add new dimensions to the application one. In [13] we show how several supplementary hierarchies are applied to domain-specific Coloured Petri Nets. This allows model elements to have at least one type from the levels above in the application hierarchy and potentially one other type per incorporated supplementary hierarchy. Although the use of supplementary hierarchies was a design choice to facilitate the addition of supplementary features to a functional main language, we are now investigating how to extend and formalize their usage for the composition of structure and behaviour of MLM hierarchies. We consider our approach as a realization of the composition process by taking advantage of the supplementary hierarchies and double typing. This is shown in Fig. 1b, where we aim to build the composition by assigning more than one type to elements in the Instance level. In this case, Hierarchy 2 is considered supplementary and its elements can be used to add additional types to elements in the Instance model.

When it comes to structure composition, we can compare the use of supplementary hier-

archies to the *Aggregation* scenario depicted in [10], where a language uses some constructs provided by other languages. With our approach, the additional languages (provided by the supplementary types) can be added/removed consistently which provides a strong separation of concerns and strengthen reusability. Hence, we use a "virtual" merge in which we achieve composition by relying on type combinations. Our goal now is to further investigate and decide which of the approaches depicted in Fig. 1 suits best.

When it comes to behaviour (MCMTs) composition, we analyse which two types (from each hierarchy) are used to double-type an element, and use this information to guide the amalgamation of the rules at runtime. The amalgamation process is based on double-typing which in turn is equivalent to type-sameness (commonality model for constructing the pushout) on which traditional merging is found.

Our next steps towards the formalization of the composition of multilevel DSMLs is twofold: (1) to determine which of the techniques shown in Fig. 1 is more convenient; and (2) to define which rules can be amalgamated, identify limitations and corner cases of the approach and determine coordination mechanism for the application of the rules.

References

[1] UML. http://www.uml.org/.

[2] C. Atkinson and T. Kühne. On evaluating multi-level modeling. In *MODELS*, 2017.

[3] F. Durán, A. Moreno-Delgado, F. Orejas, and S. Zschaler. Amalgamation of domain specific languages with behaviour. *Journal of Logical and Algebraic Methods in Programming*, 2017.

[4] S. Eker, N. Martí-Oliet, J. Meseguer, and A. Verdejo. Deduction, strategies, and rewriting. *Electronic Notes in Theoretical Computer Science*, 174(11):3–25, 2007.

[5] J. d. Lara, E. Guerra, and J. S. Cuadrado. When and how to use multilevel modelling. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 24(2):12, 2014.

[6] F. Macías. *Multilevel modelling and domain-specific languages*. PhD thesis, Western Norway University of Applied Sciences and University of Oslo, 2019.

[7] F. Macías, A. Rutle, and V. Stolz. Multecore: Combining the best of fixed-level and multilevel metamodelling. In *MULTI@ MoDELS*, pages 66–75, 2016.

[8] F. Macías, A. Rutle, V. Stolz, R. Rodriguez-Echeverria, and U. Wolter. An approach to flexible multilevel modelling. *Enterprise Modelling and Information Systems Architectures*, 13:10:1–10:35, 2018.

[9] F. Macías, U. Wolter, A. Rutle, F. Durán, and R. Rodriguez-Echeverria. Multilevel Coupled Model Transformations for Precise and Reusable Definition of Model Behaviour. *Journal of Logical and Algebraic Methods in Programming*, 2019.

[10] D. Méndez-Acuña, J. A. Galindo, T. Degueule, B. Combemale, and B. Baudry. Leveraging Software Product Lines Engineering in the development of external DSLs: A systematic literature review. *Computer Languages, Systems & Structures*, 46:206–235, 2016.

[11] J. E. Rivera, F. Durán, and A. Vallecillo. On the behavioral semantics of real-time domain specific visual languages. In *WRLA@ ETAPS*, 2010.

[12] A. Rodríguez, F. Durán, A. Rutle, and L. M. Kristensen. Executing Multilevel Domain-Specific Models in Maude. *Journal of Object Technology*, 18(2):4:1–21, 2019.

[13] A. Rodríguez, A. Rutle, L. M. Kristensen, and F. Durán. A Foundation for the Composition of Multilevel Domain-Specific Languages. In *MULTI@ MoDELS*, pages 88–97, 2019.

[14] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro. *EMF: Eclipse Modeling Framework*. Pearson Education, 2008.

[15] J. Whittle, J. Hutchinson, and M. Rouncefield. The state of practice in model-driven engineering. *IEEE software*, 31(3):79–85, 2014.