Special Issue on Multi-Level Modeling Process Challenge

Editorial

João Paulo A. Almeida[a], Thomas Kühne[*,b], Marco Montali[c]

[a] Federal University of Espírito Santo, Brazil
[b] Victoria University of Wellington, New Zealand
[c] Free University of Bozen-Bolzano, Italy

Abstract. *Multi-level modeling is an extension of traditional two-level object-oriented modeling which over the years has spawned several related solution technologies. These technologies embody a variety of multi-level modeling approaches, with differences spanning the range between superficial detail to fundamental divergence. Understanding those differences, as well as the respective trade-offs of different technologies can be difficult when looking at each technology in isolation and/or when technology demonstration application scenarios are not standardized. This EMISAJ special issue invited solutions to a multi-level modeling process challenge (Almeida et al. 2021), in order to allow contributors to showcase advantages and discuss shortcomings of their technologies. The contributions featured in this special issue therefore do not only support a deeper understanding of each technology respectively, but in combination also support comparisons among technologies.*

Keywords. Multi-Level Modeling • Process Modeling

Introduction

Multi-level modeling is an extension of traditional two-level modeling approaches (such as the UML) that has the potential to improve upon the utility, reliability and maintainability of models. Multi-level approaches allow for an unbounded number of classification levels and thus support advanced modeling concepts that foster expressiveness, reuse and controlled adaptability (Lara et al. 2014). A key aspect of the multi-level modeling paradigm is the use of entities that are simultaneously types and instances (Atkinson 1997), a feature which, in combination with further multi-level modeling concepts, has beneficial consequences for conceptual modeling, language engineering and the development of model-based software systems.

Over the years, the original potency-based multi-level modeling approach (Atkinson and Kühne 2001) has given rise to a considerable number of tools, languages, and frameworks; an overview of technologies is available at (Multi-Level Modeling Wiki 2022, Tools).

Because these technologies were predominantly developed independently from each other and targeted different priorities, today's multi-level modeling solution landscape is wide and deep. While all technologies can address typical multi-level modeling domains such as biological taxonomies, process (meta-) modeling, enforced software architectures, and systems with dynamic type levels, the particular solution mechanisms employed and the resulting trade-offs can differ considerably between certain solution technologies.

* Corresponding author.
E-mail. tk@ecs.vuw.ac.nz

Understanding what the differences between multi-level modeling technologies are, can be beneficial for

1. selecting a particular technology,
2. optimizing the use of a technology through a deeper appreciation of its particularities, and
3. driving future multi-level modeling research with respect to consolidation potential and/or need for bifurcation.

In order to support these goals, past workshops in the MULTI series (Multi-Level Modeling Wiki 2022, Events) have invited multi-level modeling solutions to so-called "modeling challenges", i. e., specific modeling scenarios with certain modeling requirements. The MULTI 2019 "Process Challenge" challenge (Almeida et al. 2019), was used as a basis for the challenge that submissions to this special issue were invited to respond to (Almeida et al. 2021). Like its predecessor, on which it is closely based, this challenge concerns the domain of process management (Dumas et al. 2005), a domain in which one is not only interested in *particular occurrences* (e. g., "processes" ≡ "process instances" or "tasks" ≡ "task occurrences"), but also in universal aspects of *classes of occurrences*, e. g., "process types", "task types" and their relations to actor types and artifact types. Furthermore, and most importantly for multi-level modeling, so-called *process metamodeling* can be used to classify this latter type level.

Model artefacts at the process metamodeling level can usefully restrict which process models can be constructed and typically provide modeling abstractions to be drawn from at lower levels. Overall, the use of multiple levels supports the separation of general domain-terminology from domains-specific counterparts, and reduces accidental complexity in comparison to two-level solutions (Atkinson and Kühne 2008).

A more in-depth account on process (meta-) modeling can be found in the "Process Challenge" description (Almeida et al. 2021) that was formulated for this special issue.

Solution Description Format

Challenge solutions had to include at least the following sections:

- Technology (precise description of the technology / approach used);
- Analysis (any disambiguations of the case description and assumptions made, any potentially added requirements);
- Model Presentation (detailed presentation of a model, including justifications for design decisions);
- Satisfaction of Requirements (demonstration of how the solution satisfies the challenge requirements);
- Assessment of the Modeling Solution (discussing choices made, pointing out potential compromises / deficiencies);
- Related Work (positioning and contrasting the presented solution with related work);
- Conclusions (including lessons learned, impulses for future work, etc.).

Evaluation Criteria

Each challenge response submitted to this special issue was reviewed against the following criteria:

i) Does the submission address the established domain as described in the challenge (Almeida et al. 2021) and does it demonstrate the use of multi-level features?

ii) Does the submission evaluate/discuss the proposed modeling solution against the challenge's criteria?

iii) Does the submission discuss the merits and limitations of the applied MLM technique in the context of the challenge?

Solutions were not required to satisfy all requirements of the challenge, however any omissions or applied workarounds had to be flagged and discussed.

In this Issue

Out of an initial set of ten expressions of interest to submit a contribution to the challenge, six materialized in submissions. In two review rounds, comprising three reviews for each submission and each round, five submissions were eventually accepted and are included in this special issue. The approach taken by the sixth submission did not facilitate easy comparisons to other solutions and will therefore be published as a regular paper.

In *Evaluating DeepTelos for ConceptBase*, Manfred Jeusfeld addresses all requirements of the challenge by employing DeepTelos (Jeusfeld and Neumayr 2016), a multi-level modeling extension of the knowledge representation language (O-)Telos (Koubarakis et al. 2021). Based on ConceptBase (Jeusfeld 2009) as an implementation platform, Jeusfeld's solution primarily builds on classification and generalization, in particular on a variant of the powertype pattern (Partridge et al. 2018). The maximum classification depth in his solution comprises four levels; or rather three, if one counts domain-specific classification relationships only, i.e., regards the so-called *omega-level* at the top as a generic language definition rather than a domain-abstraction. DeepTelos' level-agnosticism, support for multiple classification, and the ability to define user-definable modeling constructs allow Jeusfeld to address all requirements without having to employ any workarounds. Jeusfeld notes, however, that the judicious use of so-called "most general instances" may not be ideal in terms of minimizing accidental complexity. Those interested in experimenting with the DeepTelos solution may access the respective files along with instructions (Jeusfeld 2022).

In *Multi-level modeling with LML*, Arne Lange and Colin Atkinson use the Melanee tool (Gerbig 2017) which is based on the orthogonal classification architecture (Atkinson and Kühne 2003), i.e., sharply distinguishes between the use of the tool's built-in language versus the use of domain-induced levels. Melanee's Deep OCL constraint language (Lange 2016) recognizes both dimensions and has dedicated support for expressing multi-level constraints, e. g., cross-level constraints. Of further note is Melanee's strict application of classification as a level-segregation principle to the extent of prohibiting any other inter-level relationship kinds and any level-jumping. The latter limitation requires the authors to satisfy requirement S11 with what they refer to as an "unnatural" workaround, prompting them to discuss various alternative solution candidates.

In *Dual Deep Modeling of Business Processes*, Bernd Neumayr, Christoph G. Schuetz, and Michael Schrefl use their Dual Deep Modeling (DDM) approach (Neumayr et al. 2018) that features 'cross-level relationship'-supporting dual potencies. DDM elements are not confined to playing a single role at a single level and the level-segregation principle can be "concretization", i.e., more than orthodox classification. However, in their article, the authors demonstrate that when using DDM, it is possible to solely use pure classification and establish a level hierarchy in which elements have uniquely defined level-memberships. Their solution represents actors as instances of the roles they play and since DDM does not support a mechanism like multiple classification, it cannot directly support requirements P15/P16/S11. Neumayr et al. discuss three alternatives which could be used to completely satisfy the respective requirements.

In *Multi-Level Modelling with MultEcore*, Alejandro Rodríguez and Fernando Macías use their MultEcore tool (Macías 2019; Rodríguez 2022), a set of eclipse plugins that support a graphical editor and a potency-based multi-level modeling language. Their solution features a four-level process hierarchy, plus an orthogonal supplementary hierarchy. Note that MultEcore level counts are not necessarily comparable to pure classification level counts since its inter-level relationship is not restricted to classification. The authors specify static constraints and address dynamic aspects of the challenge domain by employing MultEcore's model transformation language. A bidirectional transformation is used to support model execution with the help of Maude. Since MultEcore does not support cross-level associations/links the authors had to take some care to avoid element duplication when addressing requirement S7 and could not avoid element duplication when addressing S11.

In *Multi-Level Modeling with Openflexo/FML*, Sylvain Guérin, Joel Champeau, Jean-Christophe Bach, Antoine Beugnard, Fabien Dagnat, and Salvador Martínez explore a solution based on the notion of model federation in the Openflexo/FML language (Golra et al. 2016). Since the language does not directly support multi-level modeling, the authors base their solution on the "Type Object" pattern (Johnson and Woolf 1997). Their approach can thus be understood as using conventional two-level workarounds, entailing added accidental complexity (Atkinson and Kühne 2008). Their solution uses FML code to address (level-crossing) domain requirements and requires the specification of model constraints to enforce the semantics of classification for domain types represented at the object level (e. g., in the case of the specialization of actor types). However, the use of what the authors call "virtual models" allows them to create a layered hierarchical solution, which specializes a "Base metamodel" into an "Acme metamodel", which is finally instantiated. While all challenge requirements could be satisfied by the use of FML only, two additional tools—a process type graphical editor and an enacted process graphical editor—were developed to check and validate the solution.

Acknowledgments

We would like to thank the authors for making this special issue possible by submitting their work. We are also indebted to the reviewers for their detailed and constructive reviews. João Paulo A. Almeida is supported by CNPq (313687/2020-0) and FAPES (281/2021).

About the Guest Editors

João Paulo A. Almeida is an Associate Professor at the Federal University of Espírito Santo, Brazil, and Senior Member of the Ontology and Conceptual Modeling Research Group (NEMO). His research focuses on the application of ontologies in (multi-level) conceptual modeling, enterprise architecture and enterprise modeling. He received his Ph.D. in Computer Science from the University of Twente, The Netherlands in 2006.

Thomas Kühne is an Associate Professor at Victoria University of Wellington, New Zealand. Prior to that he was an Assistant Professor at the Technische Universität Darmstadt, Germany, and an Acting Professor at the University of Mannheim, Germany. His research interests include multi-level modeling and model-driven development. He received his Ph.D. in Computer Science from the Technische Universität Darmstadt, Germany in 1998.

Marco Montali is a Full Professor and Vice-Dean for Studies at the Free University of Bozen-Bolzano, Italy, and member of Steering Committee the IEEE task force on process mining. His research is at the intersection of artificial intelligence, formal methods, and data science, for the model- and data-driven analysis of business processes and multiagent systems. He received his Ph.D. in Computer Engineering from the University of Bologna, Italy in 2009.

References

Almeida J. P. A., Kühne T., Montali M. (2021) The MULTI Process Challenge – EMISAJ Special Issue. http://purl.org/emisajchallenge

Almeida J. P. A., Rutle A., Wimmer M., Kühne T. (2019) The MULTI Process Challenge. In: 22nd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, MODELS Companion 2019, Munich, Germany, September 15-20, 2019. IEEE, pp. 164–167 https://doi.org/10.1109/MODELS-C.2019.00027

Atkinson C. (Oct. 1997) Meta-Modeling for Distributed Object Environments. In: Enterprise Distributed Object Computing. IEEE, pp. 90–101

Atkinson C., Kühne T. (Oct. 2001) The Essence of Multilevel Metamodeling. In: Proceedings of the 4^{th} International Conference on the UML 2000, Toronto, Canada. LNCS 2185. Springer Verlag, pp. 19–33

Atkinson C., Kühne T. (Sept. 2003) Model-Driven Development: A Metamodeling Foundation. In: IEEE Software 20(5), pp. 36–41

Atkinson C., Kühne T. (2008) Reducing Accidental Complexity in Domain Models. In: Software and Systems Modeling 7(3), pp. 345–359

Dumas M., van der Aalst W. M., ter Hofstede A. H. (2005) Process-aware Information Systems: Bridging People and Software Through Process Technology. John Wiley & Sons, Inc., NY, USA

Gerbig R. (2017) Deep, seamless, multi-format, multi-notation definition and use of domain-specific languages. PhD thesis https://madoc.bib.uni-mannheim.de/42010/

Golra F. R., Beugnard A., Dagnat F., Guérin S., Guychard C. (2016) Addressing Modularity for Heterogeneous Multi-model Systems using Model Federation. In: Companion Proceedings of MoMo'2016. ACM, Malaga, Spain, pp. 206–211 http://doi.acm.org/10.1145/2892664.2892701

Jeusfeld M. A. (2009) Metamodeling and method engineering with ConceptBase. In: Metamodeling for Method Engineering. MIT Press, pp. 89–168

Jeusfeld M. A. (2022) ConceptBase files for “Evaluating DeepTelos for ConceptBase”. http://conceptbase.sourceforge.net/emisaj2021challenge/

Jeusfeld M. A., Neumayr B. (2016) DeepTelos: Multi-level Modeling with Most General Instances. In: Conceptual Modeling - 35th International Conference, ER 2016, Gifu, Japan, November 14-17, 2016, pp. 198–211 https://doi.org/10.1007/978-3-319-46397-1_15

Johnson R., Woolf B. (1997) Type Object. In: Pattern Languages of Program Design 3 1st ed. Software Pattern Series. Addison-Wesley Professional chap. 4

Koubarakis M., Borgida A., Constantopoulos P., Doerr M., Jarke M., Jeusfeld M. A., Mylopoulos J., Plexousakis D. (2021) A retrospective on Telos as a metamodeling language for requirements engineering. In: Requir. Eng. 26(1), pp. 1–23 https://doi.org/10.1007/s00766-020-00329-x

Lange A. (2016) dACL: the deep constraint and action language for static and dynamic semantic definition in Melanee. MA thesis, Mannheim https://madoc.bib.uni-mannheim.de/43490/

Lara J. D., Guerra E., Cuadrado J. S. (2014) When and How to Use Multilevel Modelling. In: ACM Transactions on Software Engineering and Methodology 24(2), 12:1–12:46 https://doi.org/10.1145/2685615

Macías F. (June 2019) Multilevel modelling and domain-specific languages. PhD dissertation, Western Norway University of Applied Sciences and University of Oslo

Multi-Level Modeling Wiki. http://homepages.ecs.vuw.ac.nz/Groups/MultiLevelModeling/. Last Access: 15 March, 2022

Neumayr B., Schuetz C. G., Jeusfeld M. A., Schrefl M. (2018) Dual deep modeling: multi-level modeling with dual potencies and its formalization in F-Logic. In: Software and Systems Modeling 17(1), pp. 233–268

Partridge C., de Cesare S., Mitchell A., Odell J. (Feb. 2018) Formalization of the classification pattern: survey of classification modeling in information systems engineering. In: Software & Systems Modeling 17(1), pp. 167–203

Rodríguez A. (Jan. 2022) A Multilevel Modelling Infrastructure for the definition, execution and composition of Domain-Specific Modelling Languages. PhD dissertation, Western Norway University of Applied Sciences