A Feature-based Categorization of Multi-Level Modeling Approaches and Tools

Muzaffar Igamberdiev, Georg Grossmann, and Markus Stumptner

Advanced Computing Research Centre
School of IT and Mathematical Sciences
University of South Australia, Mawson Lakes, SA 5095, Australia
{firstname.lastname}@unisa.edu.au

Abstract. The traditional two-level modeling approaches produce accidental complexities when modeling multiple abstraction levels. This problem is addressed by the emerging multi-level modeling paradigm that allows an arbitrary number of modeling levels. To date, multiple frameworks and tool implementations have been proposed, but so far there is no comprehensive comparison of commonalities and differences of design choices that have been made in those frameworks and tools.
We propose a feature-based comparison framework that covers three core perspectives on multi-level modeling approaches and tools: language engineering, domain modeling and tool support, and evaluated a selected set of existing approaches and tools according to them.
The framework highlights research challenges and trends for future research and its results support users to choose a specific approach depending on the application domain. The framework can also be used as a basis to map and transform between approaches.

1 Introduction

Traditionally, modeling is performed within two classification levels, the model and the meta-model level. This type of modeling leads to accidental complexities as pointed out by Atkinson et al.[8]. *Multi-level modeling* supports elements and relationships across an arbitrary number of meta-levels. It overcomes the restriction of two meta-levels and supports complex domains which require a number of abstraction layers. Multi-level modeling reaches a separation of concerns in terms of linguistic and ontological dimensions by addressing linguistic and domain elements separately [8,25].

A new research area needs the firm definitions of its concepts and consensus between members of the research community to be applied in practice. Multi-level modeling is not the exception. In addition, there is a need for an objective evaluation criteria to assess the multi-level modeling approaches and tools. In this sense, a meta-discussion of the open-questions of multi-level modeling has been initiated recently [5] to achieve the required consensus and alignment of terminological differences.

This paper follows on from a previous discussion [5] but with a different goal: to propose a feature based comparison framework to categorize different multi-level modelling approaches and identify research challenges and new trends. Another goal is to to guide end users in choosing which approach is more suitable for a particular application. For example, a user with the Eclipse Modeling Framework (EMF) experience may choose Melanee [4] or the DPF Workbench [21] because of their close relationship to EMF.

The remainder of the paper is organized as follows. Section 2 lists and describes the criteria for the comparison framework. Section 3 assesses multi-level modeling approaches and tools against the criteria and highlights some results and implications. Related work is discussed in Section 4. The paper concludes with discussing limitations open challenges for multi-level modeling in Section 5.

2 Comparison criteria

The goal of the comparison criteria is to make the diversity of design choices explicit. In order to achieve this we have applied domain analysis to analyze and model the common and variable design choices or features in the context of multi-level modeling. This work was inspired by the feature-based classification of model transformations by Czarnecki et al.[11]. Differently, the framework assesses a number of approaches and tools against the criteria by means of the standard feature-diagram notation [9] to document our evaluation criteria as illustrated in Figure 1.

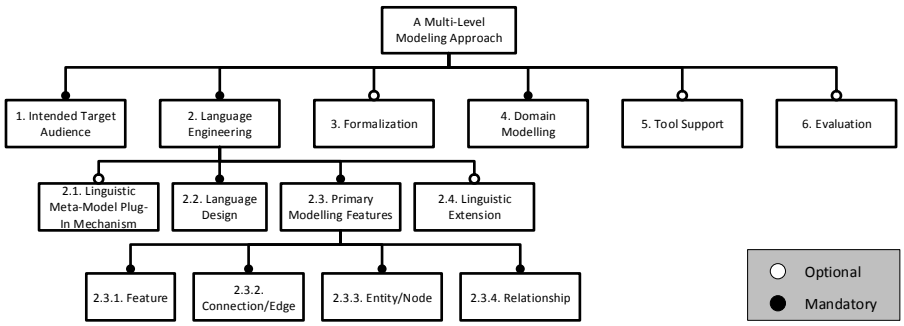

Fig. 1: Comparison criteria for multi-level modeling

Comparison criteria represent the different aspects of multi-level modeling. They are organized from the language engineering, domain modeling and tool support perspectives which are highlighted by a different background in Figure 1. Some of the criteria have been initially described in related work [25,5,15]. Each criterion is classified as *mandatory* or *optional*, for example, a multi-level modeling approach must have the language engineering, domain modeling and target audience features, while it is optional to have tool support.

1. Intended target audience. This defines the end-user, i.e., who is going to use the approach. It is important to choose the user audience to prevent misunderstandings between approaches as described in [5].

2. Language engineering. This core criterion defines the language syntax and grammar of a multi-level modeling approach.

3. Formalization. This is an important feature because it benefits an approach by avoiding the introduction of imprecision or ambiguity in the process [16]. For example, an approach can be formalized by relational logic, Diagram Predicate Framework (DPF) or F-Logic.

4. Domain modeling. This core feature deals with concepts, deep characterization and modeling features. Differently from the meta-modeling in MDE context, multi-level modeling deals with domain modeling and language engineering separately.

5. Tool support. One of the core and practical features of an approach, which demonstrates practical applicability. Sub-categories are related to domain modeling features.

6. Evaluation on industry models. This criterion identifies if the approach has been applied in big industry models. It determines the scalability, performance, and commercialization of an approach/tool among others in industry.

2.1 Language engineering perspective

This criterion organizes the features related with the linguistic dimension of a multi-level modeling approach.

2.1. Linguistic meta-model plug-in mechanism. This feature defines the flexibility to extend or to plug in the additional linguistic elements which are not addressed in the core linguistic meta-model. It may be need when an underlying model has its custom and domain related linguistic elements, such as the standards in the oil and gas industry [19].

2.2. Language design. It defines whether to define its own language or adapt a set of existing modeling concepts from model repository/library [5].

2.3. Primary modeling features. It defines the primary linguistic modeling features that an approach may have. In addition, there are the additional modeling feature, which will be addressed under the domain modeling perspective.

2.4. Linguistic extension. An ability to introduce an element at any ontological level without its ontological type [25]. It is practical to introduce an unpredictable arbitrary element.

2.3.1. Feature. It is a future of an entity (modeling element) that may be characterized by the entity's attribute and/or method. It represents respectively property and behavior of an entity.

2.3.2. Connection. It is a concept that connects two entities. It represents any type of domain related connection that relates two domain entities.

2.3.3. Entity. A main building block that represents a concept in a multi-level modeling approach.

2.3.4. Relationship. A traditional object-oriented modeling relationship to relate two entities. Differently from the connection it does not represent domain related connections.

2.2 Domain modeling perspective

This subgroup of the comparison criteria (Figure 1) addresses modeling principles of multi-level modeling. The modeling patterns and additional modeling features (Feature 4.1 and 4.4 in Figure 2) originates from [25]. These features are realized in the tool support.

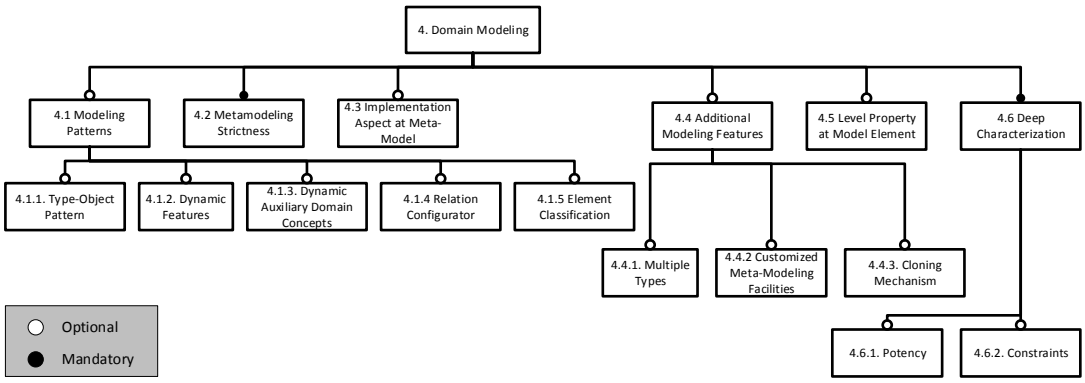

Fig. 2: Comparison criteria for domain modeling

4.1. Modeling patterns. This feature/criterion represents a group of modeling patterns, which were defined and evaluated in the meta-models from different model repositories [25], such as the OMG specifications.

4.2. Meta-modeling strictness. There are two types: strict and loose meta-modeling. The strict meta-modeling requires that only the instance-of relationship crosses exactly one meta-level boundary. In contrast, the loose meta-modeling opposes the strict one by meaning that the location of model elements is not determined by their place in the instance-of hierarchy [3].

4.3. Implementation aspect at meta-model. It defines whether the implementation details of a multi-level modeling approach are addressed at the meta-model level of the approach. It facilitates the mapping of linguistic and domain concepts into the implementation language.

4.4. Additional modeling features. This is an additional criterion to describe a group of modeling features [25] on top of the primary ones (see Feature 2.3).

4.5. Level property at model element. It represents how to number and name the ontological levels. The approaches are distinguished between the explicit and deducted (calculated) features.

4.6. Deep characterization. It defines the instantiation mechanism through ontological levels. It covers potency and constrain in multi-level models.

4.1.1. Type-object pattern. This criterion defines the explicit modeling of types and their instances [25]. The types can be added dynamically. It is useful, when one needs to add a type (e.g. a product type) dynamically, on-demand and then its instance.

4.1.2. Dynamic features. It indicates a feature that allows to add new features and their values dynamically to a type and its instances respectively [25]. It benefits when it is not possible to foresee the features needed by a certain type.

4.1.3. Dynamic auxiliary domain concepts. It is a variant of the dynamic features pattern, but, instead of features it adds dynamic entities (together with their instances) related to a type [25]. For example, an entity `Color` can be added on-demand to describe the colors of a type `Shape`.

4.1.4. Relation configurator. It allows dynamic creation of the configuration of a reference type and its instance based on that configuration [25]. For example, a shape can be restricted to have exactly one color by dynamically configuring the association ends of the reference type between an entity `Color` and a type `Shape`.

4.1.5. Element classification This feature defines the element inheritance from a dynamically created parent type to child types, consequently the element is populated in the instances of both parent and child types. The hierarchy may have some types without instances, namely abstract types. It helps to organize common features that can be reused by children [25].

4.4.1. Multiple types. This criterion characterizes an ability for a clabject to have multiple ontological types [25]. It can be useful to categorize an element according to different standards.

4.4.2. Customized meta-modeling facilities. It defines an ability to offer customization of meta-modeling features, like restriction to single inheritance or usage of a domain-specific set of data types [25].

4.4.3. Cloning mechanism. It defines the libraries with a group of predefined elements to be cloned/populated in the model being built [25]. It helps to organize the model repositories with commonly used model structures and hierarchies based on application domain.

4.6.1. Potency. It defines an instantiation mechanism by providing a number of possible instantiations through ontological levels. In each level the potency is decreased by one and eventually reaches zero at the bottommost level [7].

4.6.2. Constraints. This feature defines a constraint for entities, attributes, connections and multiplicity constraints among others.

2.3 Tool support perspective

This perspective reviews multi-level modeling against the tool support. The tool support is an essential perspective to demonstrate the abilities including practicality, performance and scalability of an approach. The tool support criteria are illustrated in Figure 3. The label of each feature is self-explanatory and is not explained here further due to space limitations.

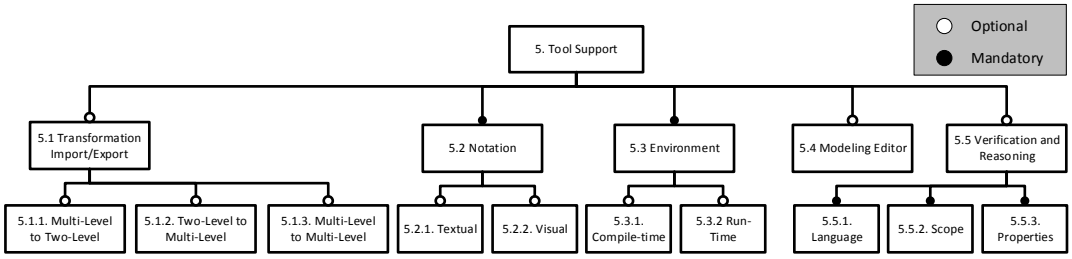

Fig. 3: Comparison criteria of tool support for multi-level modeling

3 Comparison of approaches and tools

We have selected a number of multi-level modeling approaches and tools from the literature, which will be compared against the selection criteria. The selection list is not exhaustive, though we attempted to focus on recent and popular tools. The multi-level modeling tools are organized in chronological order, starting from 1990 (Telos [27]) up to this date (DDM [32]) shown in Table 1.

Design choices. Regarding Feature 2.2 Language Design, some approaches and tools based on the existing implementation languages build their multi-level modeling language by adapting the concepts and reasoning features (e.g., DDM and OMLM), while few approaches define their own language, such as Melanee and MetaDepth. However, they also leverage on the EMF and Epsilon languages respectively. For the Feature 4.2 Meta-Modeling Strictness, only a couple of approaches [2,32] support loose, but the rest follow strict meta-modeling. Only few approaches prefer to define level number at model elements, while in others it is indicated through the potency of the model.

Challenges and trends. The Feature 2.1 Linguistic Meta-Model Plug-in Mechanism is addressed only by OMLM. This feature provides a flexibility of the linguistic meta-model extension for models with domain specific linguistic elements. The multi-level modeling can attract more attention and users by applying the approaches in real-life industry models (Feature 6), which is limited with a couple of approaches [25,19] at the moment. In terms of multi-level modeling patterns, few tools support the element classification. Addressing Feature 4.3 Implementation Aspect at Meta-Model Level provides an ability to explicitly capture the mapping of the multi-level meta-model to an existing programming language and multiple advantages such as decoupling the multi-level framework from the implementation language, comparing of implementations in different programming languages and automating code generation. Similarly, the Feature 4.4 Additional Modeling Features need more attention, while we have limited support for Feature 4.4.2 Customized Meta-Modeling Facilities.

User guidance and support. Users from two-level modeling technical space can orient their decisions based on the import and export features of the tools, so that they can import their original two-level models to benefit from multi-level modeling. Another criterion that defines usability is Feature 5.2 Tool

Notation, the visual (GUI) based tools are more attractive for users than textual notation based ones. The most of the existing verification languages in multi-level modeling tools are based on OCL, such as DeepOCL [4] and EVL [24], which attract users with OCL experience. However, non-OCL based languages are also interesting for users who are interested in futures, such as reasoning and querying facilities in Flora-2, which are not addressed in OCL based verification languages.

Application area of the comparison framework. The tool comparison can benefit to discover and exchange features between approaches and tools, to make their benefits (e.g. GUI or reasoning power) available across approaches. In addition, as we suggest in conclusion, the criteria can be a starting point for the evaluation criteria for multi-level modeling tool contest.

4 Related work

We have referred to multi-level modeling characteristics and features to build the comparison criteria. To date, several reviews of multi-level modeling approaches have analyzed multi-level modeling approaches from different perspectives [30,31,5,33,35]. A comparison presented by Atkinson et. al [5] define the minimum core criteria for an approach to be called as multi-level and deep modeling: classification relationships, type and instance dual facets and a mechanism for deep characterisation. Rossini et al. [33] define the criteria based on the expressiveness and usability of the examined modelling solutions, and compare two-level and multi-level modeling in a real life example, CloudMF, against the criteria, such as language size, OCL complexity, precision, flexibility, extensibility and tooling. Another criteria by Neumayr et al. [31] have been defined based on the making domain models more simple, concise and flexible. This criteria was extended by locality, decoupling of relationship semantics and multiple categorization by the SLICER framework to cover aspects of joint meta-models in a an interoperability scenario [35]. De Lara et al. [25] presented five multi-level modeling patterns and additional modeling features, which have been reused in our domain modeling criteria. These criteria are goal-driven. Similarly, we also define our criteria towards the research challenges and trends, and user decision support in multi-level modeling approaches and tools. Differently from these reviews, we consider the features of multi-level modeling design choices mainly from three core perspectives, language engineering, domain modeling and tool support.

5 Limitations and Conclusion

The feature hierarchy can be regarded as a course-grained representation of the domain. Its comparison results into the 'supported', 'semi-supported' and 'not supported' options only. In order to turn the comparison into an evaluation framework, we need to attach a respective weight number for each criterion to calculate the overall evaluation number of an approach or tool.

We have presented a comparison framework for multi-level modeling based on a feature hierarchy and compared the relevant approaches and tools to highlight research challenges. It supports end users in their decisions about multi-level modeling approaches and tools. To elaborate the comparison into an evaluation, a multi-level modeling tool contest, for example, in the context of the MULTI workshop [6] could provide an interesting environment for such an evaluation.

Acknowledgements

This research was supported by the South Australian Premier's Research and Industry Fund (PRIF).

Approach	Tools	2. Lang. eng.				3	4 Domain modeling												5 Tool support										6
		2.1	2.2	2.3	2.4		4.1 Mod. patterns				4.2	4.3	4.4 Add. m.			4.5	4.6 Deep		5.1	5.2	5.3		5.4	5.5 Verif				
				2.3.1			4.1.2	4.1.3	4.1.4	4.1.5			4.4.1	4.4.2	4.4.3		4.6.1	4.6.2			5.3.1	5.3.2		5.5.1	5.5.2	5.5.3		
Telos [27]	Telos		D	MA	●	●	●				L							●		T	Conceptbase	JRE	●	N	S	F		
VODAK [20]	VODAK		D	●	∼	●					L							●		T	VODAK	∼		N	A	F		
OCA [7]	Melanee		D	MA	●	●	●		●	●	S			◓		●	M	●	M2	VT	EMF	JRE	●	O	S	F		
SKIF [18]	SKIF		A	MA	●	●	●				S					N/A				T	SKIF	First-order		N	S	F		
Materialization [12]	Metaclass impl		∼	MA			●		●		S					N/A				T	∼	∼				F		
VPM [36]	VIATRA		D	MA	●	●	●				L							●	2M	V	UML	Prolog,XSB	●	N	S	F		
VMTS [26]	VMTS		A	MA		●	●				S					N/A				T	C#	.NET	●	O	S	F		
Powertype [17]				MA							L																	
DeepJava [22]	DeepJava		A	MA	●		●				S					●	M	●		T	Polyglot, javac	JRE	●			F		
Nivel [2]	Nivel		A	A	●	●	●		●		S						M	●	M2	T	Nivel	WCRL	●	N	M	F		
Aschauer et al. [1]	Traversal algorithm		D	A							L					N/A	M			T	Algorithm							
M-Objects [28]	M-SQL		A	MA	●	●	●				S					●	M			T	M-SQL	SQL		O	S	F		
Deep meta-modeling [34]	MetaDepth		D	MA	●	●	●		●	●	S			◓			SM	●	2M	T	MetaDepth	JRE		O	S	FQ		
OMME [37]	OMME		A	MA			●				S					N/A				V	EMF/Ecore	JRE						
XLM [13]	XLM		A	MA			●				S					N/A				VT	EMF	JRE	●	O	S	F		
DPF [23]	DPF workbench		A	MA		●	●				S						M	●		V	EMF	JRE	●	O	S	F		
DDI [29]	Conceptbase		A	MA		●	●		●		L			◓			M	●		T	ConceptBase	JRE	●	N	S	FQ		
DesignSpace [14]	Model Analyzer		A	MA		●		●			S					●	M	●		VT	RSM,EMF	JRE	●	O	S	F		
MLT [10]			D	MA		●					L					N/A												
OMLM [19]	OMLM, MULLER	●	A	A	●	●	●			●	S	●		◓			SM	●	2M	T	Flora-2	XSB		N	A	FQ	●	
DDM [32]	DDM		A	A	●	●	●		●	●	L			◓			SM	●		T	Flora-2	XSB		N	A	F		

Features: ● - supported, ◓ - semi-supported, empty - not supported. ∼ - unknown. N/A - not applicable.
Language engineering: 2.2 (D)efined,(A)dapted. 2.3 (M)ethod,(A)ttribute.
Domain modeling: 4.2 (S)trict, (L)oose. 4.6.1 (S)ingle, (M)ulti-potency.
Tool Support: 5.1 2M - two-level to multi-level, M2 - the opposite. 5.2 (T)extual, (V)isual. 5.5.1 (O)CL, (N)on-OCL. 5.5.2 Single & All levels. 5.5.3 (F)unctional & (Q)uality properties.

Table 1: Comparison of multi-level modeling approaches and tools.

References

1. Thomas Aschauer, Gerd Dauenhauer, and Wolfgang Pree. Representation and traversal of large clabject models. In *MODELS*, pages 17–31. Springer, 2009.
2. Timo Asikainen and Tomi Männistö. Nivel: a metamodelling language with a formal semantics. *Software & Systems Modeling*, 8(4):521–549, 2009.
3. Colin Atkinson. Supporting and applying the uml conceptual framework. In *UML*, pages 21–36. Springer, 1998.
4. Colin Atkinson and Ralph Gerbig. Melanie: multi-level modeling and ontology engineering environment. In *Proc. of 2nd Modeling Wizards*, page 7. ACM, 2012.
5. Colin Atkinson, Ralph Gerbig, and Thomas Kühne. Comparing multi-level modeling approaches. In *MULTI 2014*, page 53, 2014.
6. Colin Atkinson, Georg Grossmann, Thomas Kühne, and Juan de Lara, editors. *Proc. of MULTI Workshop co-located with ACM/IEEE MoDELS 2015*, 2015.
7. Colin Atkinson and Thomas Kühne. The essence of multilevel metamodeling. In *UML 2001*, pages 19–33. Springer, 2001.
8. Colin Atkinson and Thomas Kühne. Reducing accidental complexity in domain models. *Software & Systems Modeling*, 7(3):345–359, 2008.
9. Don Batory. *Feature models, grammars, and propositional formulas*. 2005.
10. Victorio A Carvalho, João Paulo A Almeida, Claudenir M Fonseca, and Giancarlo Guizzardi. Extending the foundations of ontology-based conceptual modeling with a multi-level theory. In *International Conference on CM*. Springer, 2015.
11. Krzysztof Czarnecki and Simon Helsen. Feature-based survey of model transformation approaches. *IBM Systems Journal*, 45(3):621–645, 2006.
12. Mohamed Dahchour, Alain Pirotte, and Esteban Zimányi. Materialization and its metaclass implementation. *IEEE TKDE*, 14(5):1078–1094, 2002.
13. Andreas Demuth, Roberto E Lopez-Herrejon, and Alexander Egyed. Cross-layer modeler: a tool for flexible multilevel modeling with consistency checking. In *Proc. of ESEC/FSE 2011*, pages 452–455. ACM, 2011.
14. Andreas Demuth, Markus Riedl-Ehrenleitner, and Alexander Egyed. Towards Flexible, Incremental, and Paradigm-agnostic Consistency Checking in Multi-level Modeling Environments. In *MULTI 2014*, page 73, 2014.
15. Sebastian Erdweg, Tijs van der Storm, Markus Völter, Laurence Tratt, Remi Bosman, William R Cook, Albert Gerritsen, Angelo Hulshout, et al. Evaluating and comparing language workbenches: Existing results and benchmarks for the future. *Computer Languages, Systems & Structures*, 44:24–47, 2015.
16. Carlos A González and Jordi Cabot. Formal verification of static software models in MDE: A systematic review. *IST*, 56:821–838, 2014.
17. Cesar Gonzalez-Perez and Brian Henderson-Sellers. A powertype-based metamodelling framework. *Software & Systems Modeling*, 5(1):72–90, 2006.
18. Patrick Hayes and Christopher Menzel. A semantics for the knowledge interchange format. In *IJCAI 2001 Workshop*, volume 1, page 145, 2001.
19. Muzaffar Igamberdiev, Georg Grossmann, Matt Selway, and Markus Stumptner. An integrated multi-level modeling approach for industrial-scale data interoperability. *Software & Systems Modeling*, pages 1–26, 2016.
20. Wolfgang Klas and Michael Schrefl. *Metaclasses and their Application: data model tailoring and database integration*. Springer Science & Business Media, 1995.
21. Ole Klokkhammer. A diagrammatic approach to deep metamodelling. Master's thesis, Department of Informatics University of Bergen, 2014.

22. Thomas Kuehne and Daniel Schreiber. Can programming be liberated from the two-level style: multi-level programming with DeepJava. In *ACM SIGPLAN Notices*, volume 42, pages 229–244. ACM, 2007.
23. Yngve Lamo, Xiaoliang Wang, Florian Mantz, Oyvind Bech, Anders Sandven, and Adrian Rutle. DPF workbench: A multi-level language workbench for MDE. *Proceedings of the Estonian Academy of Sciences*, 62(1):3–15, 2013.
24. Juan Lara and Esther Guerra. Deep meta-modelling with MetaDepth. In *TOOLS 2010*, volume LNCS 6141, pages 1–20. Springer, 2010.
25. Juan De Lara, Esther Guerra, and Jesús Sánchez Cuadrado. When and how to use multilevel modelling. *ACM TOSEM*, 24(2):12, 2014.
26. Tihamér Levendovszky, László Lengyel, Gergely Mezei, and Hassan Charaf. A systematic approach to metamodeling environments and model transformation systems in VMTS. *Electronic Notes in Theoretical Computer Science*, 2005.
27. John Mylopoulos, Alexander Borgida, Matthias Jarke, and Manolis Koubarakis. Telos: Representing Knowledge About Information Systems. *ACM TOIS*, 1990.
28. Bernd Neumayr, Katharina Grün, and Michael Schrefl. Multi-level domain modeling with M-objects and M-relationships. In *Proceedings of the Sixth Asia-Pacific Conference on Conceptual Modeling-Volume 96*, pages 107–116. ACS, 2009.
29. Bernd Neumayr, Manfred A. Jeusfeld, Michael Schrefl, and Christoph Schätz. Dual Deep Instantiation and Its ConceptBase Implementation. In *Proc. of CAiSE 2014*, LNCS 8484, pages 503–517. Springer, 2014.
30. Bernd Neumayr and Michael Schrefl. Comparison criteria for ontological multi-level modeling. In *Dagstuhl Seminar on Conceptual Modelling*. GI, 2008.
31. Bernd Neumayr, Michael Schrefl, and Bernhard Thalheim. Modeling techniques for multi-level abstraction. In *The Evolution of Conceptual Modeling*, pages 68–92. Springer, 2011.
32. Bernd Neumayr, Christoph G Schuetz, Manfred A Jeusfeld, and Michael Schrefl. Dual deep modeling: multi-level modeling with dual potencies and its formalization in F-Logic. *Software & Systems Modeling*, pages 1–36, 2016.
33. Alessandro Rossini, Juan de Lara, Esther Guerra, and Nikolay Nikolov. A comparison of two-level and multi-level modelling for cloud-based applications. In *Modelling Foundations and Applications*, pages 18–32. Springer, 2015.
34. Alessandro Rossini, Juan Lara, Esther Guerra, Adrian Rutle, and Uwe Wolter. A formalisation of deep metamodelling. *Formal Aspects of Computing*, 26(6):1115–1152, 2014.
35. Matt Selway, Markus Stumptner, Wolfgang Mayer, Andreas Jordan, Georg Grossmann, and Michael Schrefl. A conceptual framework for large-scale ecosystem interoperability. In *Conceptual Modeling*, pages 287–301. Springer, 2015.
36. Dániel Varró and András Pataricza. Vpm: A visual, precise and multilevel metamodeling framework for describing mathematical domains and uml (the mathematics of metamodeling is metamodeling mathematics). *Software and Systems Modeling*, 2(3):187–210, 2003.
37. Bernhard Volz and Stefan Jablonski. Towards an open meta modeling environment. In *Proceedings of the 10th Workshop on Domain-Specific Modeling*. ACM, 2010.