

Megamodelling in a Broad Sense

Retrospective and Outlook

Ralf Lämmel
laemmel@uni-koblenz.de
University of Koblenz
Koblenz, Germany

Jean-Marie Favre
jean-marie.favre@univ-grenoble-
alpes.fr
Université Grenoble Alpes
Grenoble, France

Vadim Zaytsev
vadim@grammarware.net
University of Twente
Enschede, The Netherlands

Abstract

We review past work on megamodelling in order to illuminate and connect different aspects and meanings of megamodelling. In our review, we track megamodelling across different technological spaces (dataware, modelware, programware, ontoware) and different scenarios of usage (such as model repositories versus technology modelling). The discussion first adopts retrospective mode and eventually switches to outlook mode on some possible developments in megamodelling. In particular, we highlight the role that knowledge resources and AI (can) play in megamodelling in a broad sense. We also submit a key challenge in the field.

CCS Concepts

• **Software and its engineering** → *System modeling languages; Architecture description languages; Orchestration languages; Software maintenance tools*; • **Computing methodologies** → *Knowledge representation and reasoning*; • **Information systems** → *Data management systems*.

Keywords

Megamodelling, Model management, Linguistic architecture, Technological space, Software language engineering, Ontologies, AI

ACM Reference Format:

Ralf Lämmel, Jean-Marie Favre, and Vadim Zaytsev. 2026. Megamodelling in a Broad Sense: Retrospective and Outlook. In *ACM/IEEE 29th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion 2026)*, October 04–09, 2026, Málaga, Spain. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3837062.3839062>

1 Introduction

Model-based development and deployment pipelines typically include automated steps such as model transformations, conformance checks and artefact generation. As these pipelines become more complex, different transformation languages may be developed; sophisticated mechanisms for synchronisation and inconsistency resolution may be introduced. Thus, it may be challenging to comprehend all artefacts and relationships involved.

This is where (classic) *megamodelling* comes into play as follows:

A megamodel is a model whose elements represent models, metamodels, languages, transformations, tools and possibly other artefacts, together with the relationships between them, such as 'conformsTo', 'transformsFromTo', or 'generates'.

Megamodelling therefore treats model management as a domain that can itself be modelled, analysed and supported systematically. The classic account presents megamodelling from the perspective of model-based and model-driven (software) engineering (MD(S)E). In the remainder of this paper, however, we adopt a broader, modernised view where the model elements in a megamodel may be drawn from more diverse categories of software artefacts and concepts in different areas of software engineering and computer science.

In the first part of this paper (Sec. 2), we offer a *retrospective*; we cover the following topics:

- (1) Transformation Composition
- (2) Model Management
- (3) Megamodels as Model Registries
- (4) Megamodelling as Modelling Foundation
- (5) Global Model Management
- (6) Co-Evolution Across Technological Spaces
- (7) Linguistic Architecture
- (8) Megamodelling in a Broad Sense
- (9) Alternative Approaches

Methodologically, we adopt a purposive, concept-centric review. Following Webster and Watson [72], we organise the literature around concepts based on our own understanding of the megamodelling space and our involvement in the development of the notion over the last approx. 20 years. The review is purposive because we selected papers for their conceptual relevance and their ability to clarify distinct meanings and nuances of 'megamodelling', rather than to achieve exhaustive coverage. In Grant and Booth's [33] terms, this study is therefore a bounded conceptual synthesis, not a systematic literature review or mapping study.

In the second part of this paper (Sec. 3), we offer an *outlook* regarding areas of active research and potential developments; we cover the following topics:

- (1) Megamodelling Meets Ontologies
- (2) Megamodelling Meets AI
- (3) Megamodelling Meets XOps

The paper is concluded in Sec. 4 with what we believe to be *the key challenge in megamodelling*, that is, the synthesis of an inclusive, but equally delimiting definition of megamodelling including its formalisation.

2 Megamodelling in Retrospect

We review the fundamentals of megamodelling, the development of megamodelling across technological spaces [43], including data-aware, modelware, and programware, and across use cases ranging from model repositories to technology modelling and beyond.

2.1 Transformation Composition

Megamodelling builds on the related concept of *transformations*, particularly the modelling and composition of transformations. Early attempts to model transformation composition date back to 1958, when UNCOL [66] introduced a notation for compiler-based transformations. This notation was later simplified into the well-known T-diagrams, or tombstone diagrams, which have long been used in compiler-construction teaching. Subsequent refinements sought to clarify the notation’s semantics; the J-notation [73], for example, was introduced to remove ambiguities. Wickerson et al. [73] provide a comprehensive account of this tradition and show how simple, high-level, language-oriented notations can remain useful for reasoning about complex transformation compositions.

2.2 Model Management

Megamodelling also builds on model-management foundations. In fact, megamodelling is considered one perspective to (or one approach to) model management (MoM). Early work emerged from the database and entity-relationship communities and addressed the management of database models through operators, mappings, change propagation, and data integration. The underlying vision dates back more than two decades [6]; its concepts, operators, semantics, and algorithms matured shortly afterwards [52, 53].

2.3 Megamodels as Model Registries

The term ‘megamodel’ first appeared in a comparatively narrow setting: a model-level graph containing models, metamodels, transformations, and, sometimes, languages as nodes and edges within an MDE or modelware context. Bézivin et al. [7] introduced the notion of megamodelling in 2003, and the term *megamodel* first appeared in a publication in 2004 [8]. There, a megamodel is a registry or model whose elements represent or refer to models and metamodels. This initial sense was especially narrow because the authors focused on the ATLAS model-management platform. Soon afterwards, Bézivin et al. [14] described ‘modelling in the large’ as management of models, metamodels, properties, and relations, with a megamodel serving as a registry for models and metamodels.

Much work on megamodelling derives from the first definition of Bézivin et al., as surveyed by Hebig et al. [36].

2.4 Megamodelling as Modelling Foundation

Megamodels are also used to describe some aspects of the foundations of modelling and MDE, for example, in work by Bézivin and Favre [12, 28–31]. Muller et al. [55] later refined these ideas in ‘*Modeling Modeling Modeling*’, whose aim was to model the activity of modelling itself.

2.5 Global Model Management

Within the Bézivin et al. school of thought, the more advanced notion of Global Model Management (GMM) subsequently emerged [1, 13, 63]. GMM combines megamodelling, other facets of model management, and operations over a network of MDE artefacts. More precisely, GMM concerns the representation, organisation, querying, manipulation, and evolution of interrelated MDE artefacts, including models, metamodels, transformations, traces, languages, tools, and services, typically through megamodels. What distinguishes GMM from generic model management is its global scope: it addresses the potentially distributed configuration space or artefact graph of an entire model-driven system, project, or platform.

2.6 Co-Evolution Across Technological Spaces

Around the time megamodelling emerged, researchers in software engineering, model-driven engineering, and related fields recognised that several communities faced similar challenges in modelling, querying, and transformation. This observation gave rise to the notion of technological spaces [43].

Evolution, and particularly the co-evolution of dependent artefacts [44], attracted substantial attention across model-driven engineering [16, 17, 71], grammarware and language engineering [40, 41], and database engineering [18, 34]. This work produced what may now be viewed as ‘small’ megamodelling patterns: artefacts and schemas before and after change, transformations at multiple levels, and supporting artefacts such as traces and deltas.

Most of this work did not explicitly invoke megamodelling. Its primary concern was usually transformation operators and their properties. In retrospect, however, it exercised megamodelling. For example, roughly a decade later, the notion of coupled transformations [44] was revisited using megamodelling explicitly to describe co-evolution patterns and their instances in repositories [45].

Favre et al. [32] also applied megamodelling directly to model evolution through model transformation. This extended the scope of megamodelling beyond registries of models, metamodels, and transformations to include their evolution over time. It also allowed model-management relations, governed by suitable operators, to stand alongside executable transformations.

Megamodelling was also applied to the change management of models; for example, a megamodel-based approach towards maintaining consistency in networks of models via bidirectional transformations in the large has been studied by Stevens [65].

The insight that distinct technological spaces involve similar consistency/transformation problems, especially in the context of bidirectional transformation across several communities, has also been reported much earlier [19], without though invoking the megamodelling notion.

2.7 Linguistic Architecture

Favre, Lämmel, and collaborators later developed the notion of linguistic architecture of software products and systems [26]. This can be understood as a form of megamodelling freed from the MDE and modelware context: its elements need not be modelling artefacts, and its relations may be both operationally and conceptually diverse. It can therefore integrate programware, grammarware, and dataware within a single megamodelling scope. Such generalised

megamodels raise challenges of formal interpretation, executability, and overlap with disciplines such as build management and reverse engineering. This line of work prompted substantial follow-up research [35, 37, 47].

Megamodelling also connects to software architecture through reverse engineering and architecture recovery. Di Rocco et al. [57], for example, use megamodels to analyse MDE projects.

Heinz et al. switch from ‘linguistic architecture’ to ‘technology modelling’ [37] when the focus shifts from system architecture to patterns of software-technology use. Technology models are neither restricted to MDE-style megamodels nor confined to modelware. Their elements may include placeholders for program and build artefacts.

2.8 Megamodelling in a Broad Sense

Linguistic architecture already represents megamodelling in a broad sense because it extends beyond modelware and supports several scenarios, including architecture description, architecture recovery, and the modelling of technology-use patterns. Other work has broadened the scope further.

Favre integrates megamodelling with the IEEE 1471 standard for software architecture, including stakeholders, views, and viewpoints [27]. Zaytsev et al. [75] use megamodelling to present a comprehensive account of parsing while abstracting from concrete technologies; they cover, for example, sequence- and tree-based artefacts and the transformations between them, such as token streams to concrete syntax trees; the authors’ broad view on megamodelling is also described elsewhere [2]. Similarly, Lämmel [45] presents megamodels for several coupled-transformation scenarios without committing to a particular technological space. In a still broader setting, Lämmel [46] uses megamodels for consistency and build management in repositories spanning several technological spaces. The connection between build management and model consistency has also been studied by Stevens [64].

2.9 Alternative Approaches

Finally, we consider work that is closely related to megamodelling in spirit but not directly connected to the research reviewed above.

In ‘models at (@) runtime’ [69, 70], the term megamodel has been adopted in a specific sense for model-driven systems whose runtime behaviour involves models.

Salay et al. [59] introduced *macromodels* as models of model-management situations in which models and transformations are connected by relations such as *submodelOf* and *refinementOf*.

Earlier work on coordinating and composing software written in languages such as Ada and JOVIAL described complex *megaprograms* comprising several programming languages and decomposed into *megamodules* [60]. This literature is difficult to transfer directly to modern megamodelling because it often fails to distinguish autonomous distributed components from their models or specifications, referring to both as megamodules [5].

In systems engineering, the term *system of systems* is used when systems thinking is applied to structures too complex to treat as single systems. Such structures may also be viewed as megamodels in a broad sense. Definitions overlap [38, 49, §1.2], but the emerging consensus distinguishes systems of systems by autonomy,

belonging, connectivity, diversity, and emergence [9]. This resonates with Bézivin et al.’s claim [3] that megamodelling implements de Rosnay’s *macroscopic* [20] as a means of managing complexity in systems of systems.

Multimodels and *multi-view modelling* are also closely related to megamodelling. Using several models together for a non-trivial modelling task makes the relations among those models a domain worthy of modelling in its own right. One example is the NGA (Node–Graph–Automaton) multimodel [74], in which an entity may be represented by a node for a high-level view, a graph for structural detail, and an automaton for executable semantics. In the context of multi-view approaches, approaches for a ‘Single Underlying Metamodel’ (SUM) [51] and Virtual SUM (V-SUM) [39] are also worth pointing out. One could say that the defining structure that relates views to the SUM is essentially a megamodel. A systematic review of multi-view software and systems modelling further shows both the diversity of view-based approaches and the persistent challenges around terminology and semantic consistency [15].

3 Megamodelling – An Outlook

We now turn to areas of active research and potential developments, with particular attention to the roles of knowledge resources and artificial intelligence in megamodelling broadly conceived.

3.1 Megamodelling Meets Ontologies

Ontologies may help to make megamodels more formal. We therefore consider their actual and potential role in model management (MoM), without restricting the discussion to megamodelling.

Early MoM research focused on operators, their semantics, and use cases such as consistency management and evolution. Ontologies were later proposed as a way to provide richer, more standardised semantics for MoM models. Several contributions illustrate this ontology-oriented direction. Matsuo et al. [50] propose an ontology-based framework for managing consistency and evolution across multiple view models. El-Gayar et al. [24] present an ontology-based model-management architecture for network-based services. Deokar et al. [21] address distributed model sharing and reuse through Semantic Web technologies. Bécan et al. [4] connect ontological knowledge to feature-model management. More recently, Ryš et al. [58] use a knowledge graph for model management in systems engineering. Suresh et al. [67] describe an ontological tool for managing mechanistic mathematical models in a pharmaceutical context. Schultz et al. [61] survey model management and propose an algebraic-specification approach. Although not ontology-based, their work likewise seeks a formal foundation for model management. Lämmel’s ‘Foundations of Software Languages’ ontology [48] is explicitly ontology-oriented and covers megamodelling-like structures across several technological spaces.

3.2 Megamodelling Meets AI

Over the past decade, machine learning and artificial intelligence have become integral to software engineering. Modern software is increasingly AI-based, while AI is also widely used to generate both conventional and AI-enabled software. This raises two questions: how can AI support megamodelling, and how can AI concepts and abstractions be incorporated into megamodelling frameworks?

A direct application of AI in the megamodelling context is retrieval or recovery of megamodels [37, 57]. Existing approaches already use information retrieval, natural-language processing, and logic-based rules. Today’s large language models (LLMs) could support a substantially more comprehensive recovery and refinement process. Related experiments in knowledge-graph engineering already suggest that LLMs can assist the construction and management of structured graph knowledge [54].

Boxology is perhaps the closest counterpart to megamodelling in the ML and AI domain. Although this work does not use the term ‘megamodelling’, its aims are closely related, and connecting the two communities could be productive. The seminal work introduced ‘boxology’ [68] using two kinds of nodes, *data* and *symbol*, and two kinds of processes, *inductive* and *deductive*, to describe 14 architectural patterns. Later research extended boxology to neuro-symbolic systems [10, 56] and more general AI systems [23, 25].

The resulting vocabulary of entities and relations has expanded. For example, Boer et al. [10] distinguish *instances* (data and symbols), *models* (statistical and semantic), *processes* (generation, transformation, and inference), and *actors* (humans, agents, and robots). Although boxologies refer naturally to models and transformations, these terms differ from their strict MDE meanings. A model may, for example, be a statistical model such as an LLM. Identifying, importing, and formalising relevant boxology concepts should therefore be a prominent item on the megamodelling research agenda.

3.3 Megamodelling Meets XOps

We use ‘XOps’ [76] here as an umbrella term for DevOps, DataOps, MLOps, and related practices. A recent literature review [76] identifies 38 XOps terms and 13 well-studied forms: AIOps, BizDevOps, CloudOps, DataOps, DevSecOps, FinOps, GitOps, MLOps, ModelOps, NetDevOps, NoOps, SecDevOps, and TwinOps.

XOps is best understood not as a strict taxonomy but as a family of disciplines for continuous lifecycle control. Each centres on a primary artefact such as code, infrastructure, data, models, security policies, cost, or reliability, and combines automation, feedback, governance, and organisational responsibility around it. Each artefact has a versioned lifecycle state that may include version, measured quality, dependencies and provenance, execution environment, and governance or approval status. This characterisation closely resembles several aspects of megamodelling, including registries, evolution, and artefact management. The need to make such dependencies explicit is particularly visible in ML systems, where hidden data dependencies, configuration and feedback relationships create substantial maintenance debt [62].

XOps therefore shares ideas with several forms of megamodelling, including megamodels as model registries [8] and megamodels for repository consistency management [46]. This raises the questions whether megamodelling and XOps could be converging on a common general concept and foundation, and, if so, what contributions megamodelling can make.

A recent addition to the XOps family is AgentOps [22], which applies XOps ideas in the context of agentic AI. It can be approximated by the following sequence of specialisations:

$$\text{DevOps} \subseteq \text{MLOps} \subseteq \text{LLMOps} \subseteq \text{AgentOps}$$

From a megamodelling perspective, the question is which modelling elements and relations can be reused across these scenarios to represent XOps forms at a foundational level and to capture recurring patterns of use. For example, in the more narrow MLOps context, Kreuzberger et al. [42] have successfully leveraged a mixed-method approach to provide an overview of the necessary principles, components, and roles, as well as the associated architecture and workflows.

4 Key Challenge

In Fig. 1, we summarise the observed progression of megamodelling from ‘collecting’ (1) models and such, towards meaningfully ‘relating’ (2) the software artefacts, towards ‘abstracting’ (3) the model elements and relationships so that, for example, patterns or architectures are modelled, towards ‘maintaining’ (4) consistency et al. among the artifacts, and prospectively towards ‘reasoning’ (5) in terms of using megamodels as a knowledge resource.

Over the more than two decades since the term megamodelling was introduced, several variations of megamodelling as well as competing notions have emerged, also in industry, including data warehouses, data lakes, DevOps, and MLOps. Software engineering has also advanced considerably with regard to management of changes, builds, etc. The challenge is no longer to demonstrate that megamodelling appears in many forms, across different communities (Sec. 2) and that it has a lot of potential for expansion (Sec. 3). We suggest the following *key challenge* in megamodelling:

Let us synthesise an inclusive and equally delimiting definition of megamodelling including its formalisation.

That is, we should focus on consolidation rather than further proliferation. The community needs a modern definition of megamodelling, a shared conceptual vocabulary, a collection of representative use cases and patterns, and a research agenda connecting formal foundations with emerging AI- and XOps-based systems. The new definition must be obviously inclusive enough to cover the survey from Sec. 2 and whatever aspects of Sec. 3 are to be regarded as megamodelling. However, broadening of megamodelling also risks making the concept excessively inclusive. Thus, we shall identify exclusion/demarcation criteria.

Megamodelling and related notions have developed into a diverse body of ideas spanning broadly across computer science. This diversity demonstrates the continuing relevance and potential of megamodelling, but it also risks leaving its concepts, terminology, and results fragmented across communities. A focused community effort such as a *Dagstuhl seminar* bringing together researchers from model management, language engineering, knowledge representation, AI, and XOps, could initiate this process and deliver tangible shared resources. This effort should also be aligned with existing research agendas. For instance, Bucchiarone et al.’s MDE research agenda [11] is relevant, as it identifies heterogeneous artefacts, repositories, scalability, runtime models, and cross-artefact links as continuing challenges.

In future work, we plan to identify the foundational principles that could underpin a unified notion of megamodelling, thereby going beyond surveying existing trends and research areas.

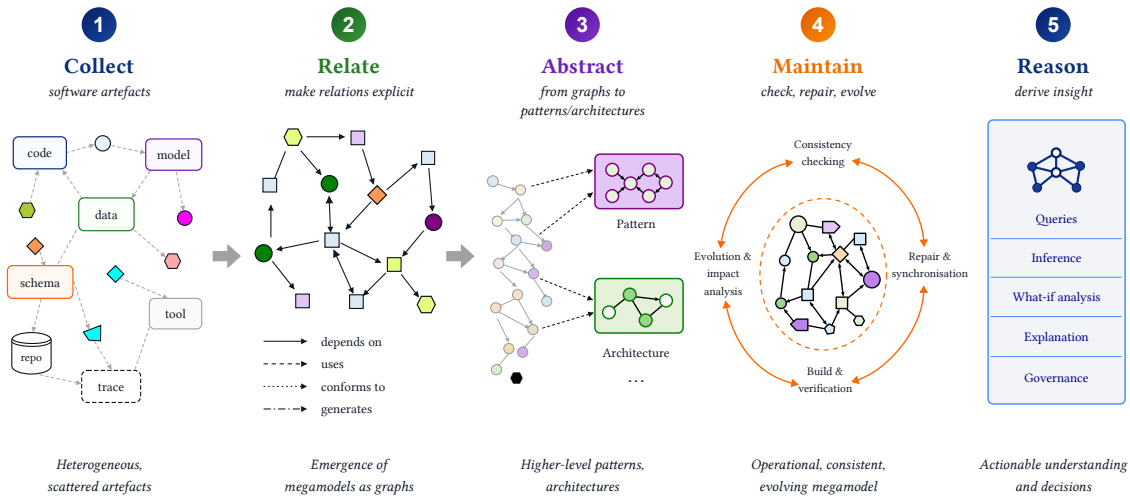


Figure 1: Stages of ‘Modern’ Megamodelling

References

- [1] Freddy Allilaire, Jean Bézivin, Hugo Brunelière, and Frédéric Jouault. 2006. Global Model Management in Eclipse GMT/AM3. In *Proc. the Eclipse Technology eXchange Workshop*. <https://inria.hal.science/hal-01272277>
- [2] Anya Helene Bagge and Vadim Zaytsev. 2015. Languages, Models and Megamodels. In *Post-proc. of SATToSE 2014 (CEUR Workshop Proceedings, Vol. 1354)*. CEUR-WS.org, 132–143.
- [3] Mikael Barbero and Jean Bézivin. 2008. Model Driven Management of Complex Systems: Implementing the Macroscopic’s vision. In *Perspectives Workshop: Model Engineering of Complex Systems (MECS) (Dagstuhl Sem. Proc., Vol. 8331)*. Schloss Dagstuhl, 1–10. doi:10.4230/DagSemProc.08331.4
- [4] Guillaume Bécan, Mathieu Acher, Benoît Baudry, and Sana Ben Nasr. 2013. *Breathing Ontological Knowledge Into Feature Model Management*. Technical Report RT-0441. INRIA, 15 pages. <https://hal.inria.fr/hal-00874867>
- [5] Dorothea Beringer, Catherine Tomabene, Pankaj Jain, and Gio Wiederhold. 1998. A Language and System for Composing Autonomous, Heterogeneous and Distributed Megamodules. In *Proceedings Ninth International Workshop on Database and Expert Systems Applications*. 826–833. doi:10.1109/DEXA.1998.707501
- [6] Philip A. Bernstein, Alon Y. Halevy, and Rachel A. Pottinger. 2000. A Vision for Management of Complex Models. *ACM SIGMOD Record* 29, 4 (Dec. 2000), 55–63. doi:10.1145/369275.369289
- [7] Jean Bézivin, Sébastien Gérard, Pierre-Alain Muller, and Laurent Rioux. 2003. MDA Components: Challenges and Opportunities. In *Workshop on Metamodelling for MDA*. 23–41. <https://hal.science/hal-00448057>
- [8] J. Bézivin, F. Jouault, and P. Valduriez. 2004. On the Need for Megamodels. In *Proc. OOPSLA/GPCE: Best Practices for Model-Driven Software Development Workshop*.
- [9] J. Boardman and B. Sausser. 2006. System of Systems - the meaning of of. In *2006 IEEE/SMC International Conference on System of Systems Engineering*. 6 pages. doi:10.1109/SYSSOE.2006.1652284
- [10] Maaike H. T. de Boer, Quirine S. Smit, Michael van Bekkum, André Meyer-Vitali, and Thomas Schmid. 2024. Modular Design Patterns for Generative Neuro-Symbolic Systems. In *ESWC*. CEUR.
- [11] Antonio Bucchiarone, Jordi Cabot, Richard F. Paige, and Alfonso Pierantonio. 2020. Grand challenges in model-driven engineering: an analysis of the state of the research. *Softw. Syst. Model.* 19, 1 (2020), 5–13. doi:10.1007/S10270-019-00773-6
- [12] Jean Bézivin. 2005. On the Unification Power of Models. *SoSyM* 4, 2 (2005), 171–188.
- [13] Jean Bézivin, Jean-Marie Favre, and Bernhard Rumpe. 2006. Introduction to GaMMA 2006: First International Workshop on Global Integrated Model Management. In *Proc. 2006 International Workshop on Global Integrated Model Management (GaMMA ’06)*. ACM, 1–3. doi:10.1145/1138304.1138305
- [14] Jean Bézivin, Frédéric Jouault, Peter Rosenthal, and Patrick Valduriez. 2005. Modeling in the Large and Modeling in the Small. In *European MDA Workshops MDAFA 2003 and MDAFA 2004, Revised Selected Papers (LNCS, Vol. 3599)*. Springer, 33–46.
- [15] Antonio Cicchetti, Federico Ciccozzi, and Alfonso Pierantonio. 2019. Multi-view approaches for software and system modelling: a systematic literature review. *Softw. Syst. Model.* 18, 6 (2019), 3207–3233. doi:10.1007/S10270-018-00713-W
- [16] Antonio Cicchetti, Davide Di Ruscio, Romina Eramo, and Alfonso Pierantonio. 2008. Automating Co-evolution in Model-Driven Engineering. In *Proc. ECOC*. IEEE, 222–231.
- [17] Antonio Cicchetti, Davide Di Ruscio, and Alfonso Pierantonio. 2007. A Metamodel Independent Approach to Difference Representation. *J. Object Technol.* 6, 9 (2007), 165–185. doi:10.5381/jot.2007.6.9.a9
- [18] Anthony Cleve and Jean-Luc Hainaut. 2006. Co-transformations in Database Applications Evolution. In *GTTSE 2005, Revised Papers (LNCS, Vol. 4143)*. Springer, 409–421.
- [19] Krzysztof Czarnecki, J. Nathan Foster, Zhenjiang Hu, Ralf Lämmel, Andy Schürr, and James F. Terwilliger. 2009. Bidirectional Transformations: A Cross-Discipline Perspective. In *Proc. Theory and Practice of Model Transformations - 2nd International Conference (Lecture Notes in Computer Science, Vol. 5563)*. Springer, 260–283. doi:10.1007/978-3-642-02408-5_19
- [20] Joël de Rosnay. 1979. *The Macroscopic: A New World Scientific System*. Harper & Row.
- [21] Amit V. Deokar and Omar F. El-Gayar. 2009. Enabling Distributed Model Management Using Semantic Web Technologies. In *Proceedings of the 42nd Hawaii International Conference on System Sciences*. IEEE, 1–9. doi:10.1109/HICSS.2009.190
- [22] Liming Dong, Qinghua Lu, and Liming Zhu. 2024. AgentOps: Enabling Observability of LLM Agents. arXiv:2411.05285 [cs.AI] <https://arxiv.org/abs/2411.05285>
- [23] Fajar J. Ekaputra, Alexander Prock, and Elmar Kiesling. 2025. Towards Supporting AI System Engineering with an Extended Boxology Notation. In *KG-STAR 2025 Knowledge Graphs for Responsible AI 2025 (CEUR, Vol. 4018)*. CEUR Workshop Proceedings.
- [24] Omar F. El-Gayar and Amit V. Deokar. 2012. An Ontology-Based Model Management Architecture for Service Innovation. In *Modeling for Decision Support in Network-Based Services*. Lecture Notes in Business Information Processing, Vol. 42. Springer-Verlag, 143–168. doi:10.1007/978-3-642-27612-5_7
- [25] Alexis Ellis, Brandon Dave, Hugh Salehi, Subhashini Ganapathy, and Cogan Shimizu. 2024. EASY-AI: sEmantic And compoSable gLyphs for representing AI systems. In *HHAI 2024: Hybrid Human AI Systems for the Social Good - Proceedings of the 3rd International Conference on Hybrid Human-Artificial Intelligence (Frontiers in Artificial Intelligence and Applications)*. 105–113. doi:10.3233/FAIA240187
- [26] Jean-Marie Favre, Ralf Lämmel, and Andrei Varanovich. 2012. Modeling the Linguistic Architecture of Software Products. In *Proc. MODELS (LNCS, Vol. 7590)*. Springer, 151–167.
- [27] Jean-Marie Favre. 2004. CacOphoNy: Metamodel-Driven Architecture Recovery. In *Proc. of the 11th Working Conference on Reverse Engineering (WCRE ’04)*. IEEE Computer Society.
- [28] Jean-Marie Favre. 2004. Towards a Basic Theory to Model Model Driven Engineering. In *Proceedings of the 3rd Workshop in Software Model Engineering (WiSME)*. 262–271. https://www.researchgate.net/publication/239016408_Towards_a_Basic_Theory_to_Model_Model_Driven_Engineering
- [29] Jean-Marie Favre. 2005. Foundations of Meta-pyramids: Languages vs. Metamodels – Episode II: Story of Thotus the Baboon. In *Language Engineering for Model-Driven Software Development (Dagstuhl Seminar Proceedings, 04101)*.
- [30] Jean-Marie Favre. 2005. Foundations of Model (Driven) (Reverse) Engineering: Models – Episode I: Stories of The Fidus Papyrus and of The Solarus. In *Language Engineering for Model-Driven Software Development (Dagstuhl Seminar Proceedings, 04101)*.
- [31] Jean-Marie Favre. 2006. Megamodelling and Etymology. In *Transformation Techniques in Software Engineering (Dagstuhl Sem. Proc., Vol. 5161)*. Schloss Dagstuhl.

- doi:10.4230/DagSemProc.05161.6
- [32] Jean-Marie Favre and Tam Nguyen. 2005. Towards a Megamodel to Model Software Evolution Through Transformations. *Electronic Notes in Theoretical Computer Science* 127, 3 (2005), 59–74. doi:10.1016/j.entcs.2004.08.034
 - [33] Maria J. Grant and Andrew Booth. 2009. A Typology of Reviews: An Analysis of 14 Review Types and Associated Methodologies. *Health Information and Libraries Journal* 26, 2 (2009), 91–108. doi:10.1111/j.1471-1842.2009.00848.x
 - [34] Jean-Luc Hainaut. 2006. The Transformational Approach to Database Engineering. In *GTSE 2005, Revised Papers (LNCS, Vol. 4143)*. Springer, 95–143. doi:10.1007/11877028_4
 - [35] Johannes Härtel, Lukas Härtel, Marcel Heinz, Ralf Lämmel, and Andrei Varanovich. 2017. Interconnected Linguistic Architecture. *The Art, Science, and Engineering of Programming Journal* 1 (2017), Issue 1. doi:10.22152/programming-journal.org/2017/1/3
 - [36] Regina Hebig, Andreas Seibel, and Holger Giese. 2011. On the Unification of Megamodels. In *Proceedings of the 4th International Workshop on Multi-Paradigm Modeling (EC-EASST, Vol. 42)*. doi:10.14279/tuj.eceasst.42.704
 - [37] Marcel Heinz, Johannes Härtel, and Ralf Lämmel. 2020. Reproducible Construction of Interconnected Technology Models for EMF Code Generation. *J. Object Technol.* 19, 2 (2020), 8:1–25. doi:10.5381/JOT.2020.19.2.A8
 - [38] M. Jamshidi. 2017. *Systems of Systems Engineering: Principles and Applications*. CRC Press.
 - [39] Heiko Klare, Max E. Kramer, Michael Langhammer, Dominik Werle, Erik Burger, and Ralf Reussner. 2021. Enabling consistency in view-based system development - The Vitruvius approach. *J. Syst. Softw.* 171 (2021), 110815. doi:10.1016/j.jss.2020.110815
 - [40] Paul Klint, Ralf Lämmel, and Chris Verhoef. 2005. Toward an Engineering Discipline for Grammarware. *ACM Trans. Softw. Eng. Methodol.* 14, 3 (2005), 331–380. doi:10.1145/1072997.1073000
 - [41] Jan Kort and Ralf Lämmel. 2003. A Framework for Datatype Transformation. In *Workshop on Language Descriptions, Tools and Applications (LDTA@ETAPS) (ENTCS, Vol. 52)*. Elsevier, 463–482. doi:10.1016/S1571-0661(05)82623-4
 - [42] Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirsch. 2023. Machine Learning Operations (MLOps): Overview, Definition, and Architecture. *IEEE Access* 11 (2023), 31866–31879. doi:10.1109/ACCESS.2023.3262138
 - [43] Ivan Kurtev, Jean Bézin, and Mehmet Akşit. 2002. Technological Spaces: An Initial Appraisal. In *Proc. CoopIS, DOA 2002, Industrial track*. <https://research.utwente.nl/en/publications/technological-spaces-an-initial-appraisal/>
 - [44] Ralf Lämmel. 2004. Coupled Software Transformations. In *Proc. SET*. 31–35. Extended Abstract. Available at <http://post.queensu.ca/~zouy/files/set-2004.pdf#page=38>.
 - [45] Ralf Lämmel. 2016. Coupled Software Transformations Revisited. In *Proc. SLE*. ACM, 239–252. doi:10.1145/2997364.2997366
 - [46] Ralf Lämmel. 2017. Relationship Maintenance in Software Language Repositories. *The Art, Science, and Engineering of Programming Journal* 1 (2017), Issue 1. doi:10.22152/programming-journal.org/2017/1/4
 - [47] Ralf Lämmel and Andrei Varanovich. 2014. Interpretation of Linguistic Architecture. In *Proc. ECMFA (LNCS, Vol. 8569)*. Springer, 67–82. doi:10.1007/978-3-319-09195-2_5
 - [48] Ralf Lämmel. 2026. Towards an Ontology for the Foundations of Software Languages. arXiv:2605.17374 [cs.SE] <https://arxiv.org/abs/2605.17374>
 - [49] Mark W. Maier. 1998. Architecting Principles for Systems-of-Systems. *Systems Engineering* 1, 4 (1998), 267–284. doi:10.1002/(SICI)1520-6858(1998)1:4<267::AID-SYS3>3.0.CO;2-D
 - [50] Atsushi Matsuo, Kazuhisa Seta, Yusuke Hayashi, and Mitsuru Ikeda. 2004. An Ontological Framework for Evolutionary Model Management. In *Conceptual Modeling for Advanced Application Domains (LNCS, Vol. 3289)*. Springer, 363–376. doi:10.1007/978-3-540-30466-1_34
 - [51] Johannes Meier, Christopher Werner, Heiko Klare, Christian Tunjic, Uwe Aßmann, Colin Atkinson, Erik Burger, Ralf Reussner, and Andreas Winter. 2019. Classifying Approaches for Constructing Single Underlying Models. In *Model-Driven Engineering and Software Development - 7th International Conference, MODELSWARD 2019, Prague, Czech Republic, February 20–22, 2019, Revised Selected Papers (Communications in Computer and Information Science, Vol. 1161)*. Springer, 350–375. doi:10.1007/978-3-030-37873-8_15
 - [52] Sergey Melnik. 2004. *Generic Model Management: Concepts and Algorithms*. Lecture Notes in Computer Science, Vol. 2967. Springer. doi:10.1007/b97859
 - [53] Sergey Melnik, Philip A. Bernstein, Alon Halevy, and Erhard Rahm. 2004. *A Semantics for Model Management Operators*. Technical Report MSR-TR-2004-59. Microsoft Research. <https://www.microsoft.com/en-us/research/publication/a-semantics-for-model-management-operators/>
 - [54] Lars-Peter Meyer, Claus Stadler, Johannes Frey, Norman Radtke, Kurt Junghanns, Roy Meissner, Gordian Dziwis, Kirill Bulert, and Michael Martin. 2023. LLM-assisted Knowledge Graph Engineering: Experiments with ChatGPT. In *First Working Conference on Artificial Intelligence Development for a Resilient and Sustainable Tomorrow - AI Tomorrow 2023, Leipzig, Germany, 29-30 June, 2023 (Informatik Aktuell)*. Springer, 103–115. doi:10.1007/978-3-658-43705-3_8
 - [55] Pierre-Alain Muller, Frédéric Fondement, and Benoît Baudry. 2012. Modeling Modeling Modeling. *Software & Systems Modeling* 11 (July 2012). doi:10.1007/s10270-010-0172-x
 - [56] C. H. Ng, Annette ten Teije, and Frank van Harmelen. 2025. A Boxology-Based Analysis of Design Patterns for Neuro-Symbolic Medical Decision Making Systems. In *Proceedings of the 23rd International Conference on Artificial Intelligence in Medicine (AIME), Part I (LNCS/LNAI, Vol. 15734)*. Springer, 333–343. doi:10.1007/978-3-031-95838-0_33
 - [57] Juri Di Rocco, Davide Di Ruscio, Johannes Härtel, Ludovico Iovino, Ralf Lämmel, and Alfonso Pierantonio. 2020. Understanding MDE Projects: Megamodels to the Rescue for Architecture Recovery. *Softw. Syst. Model.* 19, 2 (2020), 401–423. doi:10.1007/S10270-019-00748-7
 - [58] Arkadiusz Ryś, Lucas Lima, Joeri Exelmans, Dennis Janssens, and Hans Vangheluwe. 2025. Model Management to Support Systems Engineering Workflows Using Ontology-Based Knowledge Graphs. arXiv:2512.09596 [cs.SE] <https://arxiv.org/abs/2512.09596>
 - [59] Rick Salay, John Mylopoulos, and Steve Easterbrook. 2009. Using Macromodels to Manage Collections of Related Models. In *Advanced Information Systems Engineering*. Springer, 141–155. doi:10.1007/978-3-642-02144-2_15
 - [60] Neal Sample, Dorothea Beringer, Laurence Melloul, and Gio Wiederhold. 1999. CLAM: Composition Language for Autonomous Megamodels. In *Coordination Languages and Models*. Springer Berlin Heidelberg, 291–306. doi:10.1007/3-540-48919-3_21
 - [61] Patrick Schultz, David I. Spivak, and Ryan Wisnesky. 2023. Algebraic Model Management: A Survey. *CoRR* abs/2301.04846 (2023). arXiv:2301.04846 [cs.LO] <https://arxiv.org/abs/2301.04846>
 - [62] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. 2015. Hidden Technical Debt in Machine Learning Systems. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7–12, 2015, Montreal, Quebec, Canada*. 2503–2511. <https://proceedings.neurips.cc/paper/2015/hash/86df7dcfd896caf2674f757a2463eba-Abstract.html>
 - [63] Andreas Seibel, Stefan Neumann, and Holger Giese. 2010. Dynamic Hierarchical Mega Models: Comprehensive Traceability and Its Efficient Maintenance. *Software and Systems Modeling* 9, 4 (2010), 493–528. doi:10.1007/s10270-009-0146-z
 - [64] Perdita Stevens. 2020. Connecting software build with maintaining consistency between models: towards sound, optimal, and flexible building from megamodels. *Softw. Syst. Model.* 19, 4 (2020), 935–958. doi:10.1007/S10270-020-00788-4
 - [65] Perdita Stevens. 2020. Maintaining consistency in networks of models: bidirectional transformations in the large. *Softw. Syst. Model.* 19, 1 (2020), 39–65. doi:10.1007/S10270-019-00736-X
 - [66] J. Strong, J. Wegstein, A. Titter, J. Olszyn, O. Mock, and T. Steel. 1958. The Problem of Programming Communication with Changing Machines: A Proposed Solution (part 2). *Commun. ACM* 1, 9 (sep 1958), 9–15. doi:10.1145/368892.368915
 - [67] P. Suresh, S.-H. Hsu, P. Akkisetty, G. V. Reklaitis, and V. Venkatasubramanian. 2010. OntoMODEL: Ontological Mathematical Modeling Knowledge Management in Pharmaceutical Product Development. 1: Conceptual Framework. *Industrial & Engineering Chemistry Research* 49, 17 (2010), 7758–7767. doi:10.1021/ie100246w
 - [68] Frank van Harmelen and Annette ten Teije. 2019. A Boxology of Design Patterns for Hybrid Learning and Reasoning Systems. *CoRR* abs/1905.12389 (2019). <http://arxiv.org/abs/1905.12389>
 - [69] Thomas Vogel and Holger Giese. 2014. Model-Driven Engineering of Self-Adaptive Software with EUREMA. *ACM Transactions on Autonomous and Adaptive Systems* 8, 4, Article 18 (2014), 18:1–18:33 pages. doi:10.1145/2555612
 - [70] Thomas Vogel, Andreas Seibel, and Holger Giese. 2011. The Role of Models and Megamodels at Runtime. In *Models in Software Engineering (LNCS, Vol. 6627)*. Springer, 224–238. doi:10.1007/978-3-642-21210-9_22
 - [71] Guido Wachsmuth. 2007. Metamodel Adaptation and Model Co-adaptation. In *Proc. ECOOP (LNCS, Vol. 4609)*. Springer, 600–624. doi:10.1007/978-3-540-73589-2_28
 - [72] Jane Webster and Richard T. Watson. 2002. Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Quarterly* 26, 2 (2002), xiii–xxiii. <https://www.jstor.org/stable/4132319>
 - [73] John Wickerson. 2020. Diagrams for Composing Compilers. Wickopedia (personal blog), <https://johnwickerson.wordpress.com/2020/05/21/diagrams-for-composing-compilers/>
 - [74] Vadim Zaytsev. 2017. Megamodelling with NGA Multimodels. In *CoCoS*. ACM, 1–6. doi:10.1145/3141842.3141843
 - [75] Vadim Zaytsev and Anya Helene Bagge. 2014. Parsing in a Broad Sense. In *Proc. MODELS (LNCS, Vol. 8767)*. Springer, 50–67. doi:10.1007/978-3-319-11653-2_4
 - [76] Xinrui Zhang, Pincan Zhao, and Jason Jaskolka. 2025. Navigating the DevOps landscape. *Journal of Systems and Software* 223 (2025), 112331. doi:10.1016/j.jss.2024.112331