SoSyM manuscript No.
(will be inserted by the editor)

Grand Challenges in Model-Driven Engineering

An analysis of the state of the research

Antonio Bucchiarone · Jordi Cabot · Richard F. Paige · Alfonso Pierantonio

Received: date / Accepted: date

Abstract In 2017 and 2018, two events were held – in Marburg, Germany, and San Vigilio di Marebbe, Italy, respectively – focusing on an analysis of the state-of-research, state-of-practice and state-of-the-art in Model-Driven Engineering (MDE). The events brought together experts from industry, academia, and the open-source community to assess what has changed in research in MDE over the last ten years, what challenges remain, and what new challenges have arisen. This article reports on the results of those meetings, and presents a set of *grand challenges* that emerged from discussions and synthesis. These challenges could lead to research initiatives for the community going forward.

Keywords Model-driven engineering · Grand challenge · Research roadmap

1 Introduction

The field of Model-Driven Engineering [1] (MDE) has evolved substantially from the earliest work on UML in the 1990s, through to seminal research on metamodelling, model transformation, and model management in the early-to-mid 2000s.

Antonio Bucchiarone
Fondazione Bruno Kessler
Trento, Italy
E-mail: bucchiarone@fbk.eu

Jordi Cabot
ICREA and UOC
Barcellona, Spain
E-mail: jordi.cabot@icrea.cat

Richard F. Paige
University of York and McMaster University
York, UK and Hamilton, Canada
E-mail: paigeri@mcmaster.ca

Alfonso Pierantonio
Università degli Studi di L'Aquila
L'Aquila, Italy
E-mail: alfonso.pierantonio@univaq.it

MDE has made incredible contributions to leverage abstraction and automation in almost every area of software and systems development and analysis. In many domains, including railway systems, automotive, business process engineering, and embedded systems, models are key to success in modern software engineering processes. However, this success has led to an even higher demand for better tools, theories, and general awareness about modeling, its scope, and application. The changing face of MDE is reflected in the surveys (both broad and specific), roadmaps, and research challenge workshops found in the literature [2,3,4,5].

In 2017 and 2018, the research community held two events: the *Grand Challenges in MDE* workshop[1], co-located with STAF 2017 in Marburg, Germany in July 2017; and the *Winter Modeling Meeting*[2], held in San Vigilio di Marebbe, Italy, in January 2018. Experts from industry and academia attended these meetings and presented their views that reflected on the research roadmaps of the past, and the challenges facing the community in the future. This article attempts to synthesize the discussion at these two meetings. Moreover, it outlines how far the community has come in addressing challenges presented in previously published research roadmaps (particularly [4,3]), and what new challenges have arisen in the intervening years. The meetings were attended by an overlapping set of experts with different backgrounds and experience. The Grand Challenges in MDE 2017 workshop was a traditional workshop with paper submissions and presentations, along with extensive discussion. The Winter Modeling Meeting 2018 was an invitation-only workshop with focused sessions on research challenges as well as educational challenges facing MDE. The interested reader can find lists of participants and papers for these workshop on the aforementioned websites.

This article attempts to summarize the discussion of world experts in MDE at these two venues, capturing a vision of the grand challenges facing the community. As such, it may provide useful context for future research projects, research grant proposals, or presentations made to funding bodies. The paper starts with a brief reflection on research challenges identified in previous roadmaps, in order to contextualize the new challenges. Then, the paper summarizes the key challenges identified during discussions at the Grand Challenges in MDE 2017 workshop and the Winter Modeling Meeting 2018. It then concludes with a brief summary of where the authors believe the field of MDE research is going.

2 Analysis of past challenges

There has been substantial progress in research on MDE since the late 1990s and early 2000s. This was analyzed, and the state-of-the-art synthesized, in a selection of research roadmaps and challenge papers published at the time. In this section, we briefly reflect on past research challenges in MDE, in order to contextualize the results of the two workshops.

[1] `http://www.edusymp.org/Grand2017/en`

[2] `http://eventmall.info/AMM2018/`

2.1 Pre-2007 Challenges

The period from the late 1990s through to around 2007 was dominated by modelling language issues. This was a time where the Unified Modeling Language (UML) was undergoing considerable changes to its semantics, infrastructure and superstructure, and there was a very substantial body of research considering precise semantics of such modeling languages, as well as the metamodeling process [6, 7, 8, 9]. The key use case for modeling was code generation, as embodied by research on model-to-text transformation languages [10] and standards[3], and the popularity of code generators that were offered "out of the box" in modeling tools, such as the Kennedy Carter (now Abstract Solutions)[4] or Artisan (now PTC Integrity Modeler)[5] tools.

This was also a period with substantial effort in standardization, which led to the production of the Meta-Object Facility (MOF)[6], Model-Driven Architecture (MDA)[7], the Object Constraint Language (OCL)[8], and Query-View-Transformation (QVT)[9] specifications. Researchers in modeling engaged in a significant way with relevant standardization efforts, with varying degrees of success. In essence, this period laid the groundwork for more recent research, providing the foundations needed for more advanced research on modeling tools and model management.

2.2 Challenges from 2007 through present day

More recently, various research roadmaps [11, 4, 5] identified a variety of significant research challenges, many of which have seen substantial research effort. The key issues that were identified in these previously published roadmaps include the following

- *Language engineering* (e.g., [12]): principles, practices and patterns for specifying abstract and concrete syntax, as well as semantics. Research challenges related to understanding the language engineering process were also identified.
- *Language workbenches* (popularized in 2005) – i.e., tools for defining and composing domain-specific languages and their IDEs: the fundamental research against this challenge led to the development of modern language workbenches such as JetBrains MPS[10], Xtext[11], Kermeta[12], Racket[13] and Spoofax[14].
- *Model management* – processes and tasks for manipulating and analyzing models: the fundamental research in this area led to theoretical results (e.g., identification of different model management tasks, such as model merging and

[3] https://www.omg.org/spec/MOFM2T/1.0/
[4] https://abstractsolutions.co.uk/our-services/executable-uml/
[5] https://www.ptc.com/en/products/plm/plm-products/integrity-modeler
[6] https://www.omg.org/mof/
[7] https://www.omg.org/mda/
[8] https://www.omg.org/spec/OCL/About-OCL/
[9] https://www.omg.org/spec/QVT/About-QVT/
[10] https://www.jetbrains.com/mps/
[11] https://www.eclipse.org/Xtext/
[12] http://www.kermeta.org/
[13] https://racket-lang.org
[14] http://strategoxt.org/Spoofax

comparison) as well as technical contributions (e.g., model management platforms such as AtlanMod[15] and Epsilon[16]).
- *Model analysis*: the challenge of techniques for analyzing models (e.g., for performance or correctness [13]), along with principles relate to understanding what makes a good model.
- *Models at run-time*: the use of models to manage and understand systems after they have been deployed and as they execute behaviour [14]. Substantial research has taken place regarding this challenging to identify techniques and tools for automatically reflecting changes from a system into changes in models, and vice versa. This particular challenge is at the intersection of modeling and artificial intelligence research.
- *Modeling repositories* (such as REMODD [15], the Atlantic Zoo[17], and MDEForge [16]), which provide persistence for modeling artefacts such as models, transformations and metamodels: there was an identified need for not only more modeling artefacts to support research, but facilities to make it easier for engineers to store and acquire such artefacts. The adoption of such repositories is sporadic in the community.
- *Scalability across different dimensions*: given progress against some of the other challenges listed above, the ambition of researchers and engineers increased. As a result, demand for support for working with very large models (with hundreds of thousands of elements, if not more), large metamodels, large transformations etc., increased [17]. This in turn led to fundamental work on understanding the performance of modeling infrastructure, on fragmenting and splitting large models and metamodels, and on scheduling the execution of transformations to optimize their performance.

Substantial progress was made in these areas over the last period of time, and active research continues against many of these areas. These challenges fed in to the discussion sessions at the Grand Challenges in MDE 2017 workshop, and the Winter Modeling Meeting in 2018, as we now discuss.

3 Technical Challenges

This section describes the technical challenges discussed in both events. We split them up into *foundation*, *domain*, and *tool* challenges, respectively. The categorization is not strict since it does not have crispy boundaries; on the contrary, it is a pragmatic one and aims at facilitating the presentation of the challenges. As a consequence, some challenges necessarily span more than one category.

The majority of the presented challenges are of technical nature, but, as the MDE ecosystem matures and the technical issues are addressed, we believe the social and community challenges will become the critical factors for the success of MDE. The next sections shed some more light on these aspects.

[15] `http://www.atlanmod.org/`
[16] `http://www.eclipse.org/epsilon`
[17] `https://www.imt-atlantique.fr/fr`

3.1 Foundation Challenges

The foundation dimension comprehends all the challenges concerning conceptual and theoretical aspects of MDE, covering all the phases of software development (i.e., modeling, deployment, execution, and maintenance). Modeling is a well-established and successful discipline that has been practiced for decades. As a consequence, there might be good reasons for which companies want to exploit these (long-lived) models posing the question about how do we allow *legacy models* (and hence legacy modeling formats) to remain in existence. Supporting such tasks can take advantage of modeling itself by allowing legacy models to co-exist with modern modeling technologies. In this context, *agile and lean software development* is increasingly adopted in the software industry. No matter of fact: this is changing the way software is described. Companies must not move away from modeling, making model-driven development valuable at the age of agile development.

As we will see in the domain challenges section, there is a compelling need to improve the MDE solutions in order to support those processes that intrinsically include also social aspects as in multi-disciplinarity and heterogeneous environments. Thus, proper model management is an increasingly pressing challenge. In this setting, *How to transform a software engineer into a system engineer* that must be able to combine different types of models leads to an integrated view on a system?. How can we virtualize these complex systems that are based on a collection of heterogeneous models?

In systems running in an open environment (i.e., *Smart* systems*), uncertainty during the design of software models is caused by many design alternatives, incomplete information, conflicting stakeholder opinions. How to use MDE to *(i)* connect discussion models with software artifacts, *(ii)* relate different models to different choices, *(iii)* detect proposed solutions for each choice, *(iv)* learn a Design Space Exploration specification from proposed solution examples, *(v)* support fuzzy/naturalistic argumentation, *(vi)* leverage/integrate flexible modelling tools, are all needed aspects to take into account.

Considering the runtime phase of such systems, and the adaptive nature of most of the complex systems developed in the last years, we can say that *software changes are ubiquitous and unavoidable.* To manage them, we need to go towards a theory of *software agility in MDE* able to consider different kinds of maintenance, including repair and improvement, adaptation to a new platform, extension with new functionality, reuse in different contexts, refactoring to make the above kinds of maintenance more accessible). At the same time, we need to introduce theories and techniques able to detect/predict software anomalies and suggest the needed software evolutions.

To automatize and make more powerful all the maintenance solutions, we need to extend the MDE techniques exploiting AI techniques that nowaday are ready to be used for complex and highly dynamic systems. MDE techniques can help in the improvement of AI, Machine learning, and other cognification techniques. At the same time, cognification techniques can be exploited to improve and bring quantifiable and perceivable advantages to MDE solutions [18]. Machine learning is a technique that builds on the premise of having a tremendous impact not only on the way software behaves and is realized, but also on society. However, its adoptions requires massive skill sets that current professional profiles fail to

meet despite the increasing demand. Machine learning practice would be easier if the learning curve for the needed skills would be more convenient. Model-driven software engineering and human-computer interaction design can help in abstracting machine learning technology and, starting from these abstractions, enabling automated code generation.

Due to the complexity of the targeted systems, there is a strong need to increase the usability of the *Model Transformation techniques*. Model Transformations are cornerstone components of any project adopting model-driven techniques, particularly model-to-model model transformations. Current transformation languages, e.g., ATL, QVT, ETL, Henshin, VIATRA, and Stratego, provide rather powerful features and useful capabilities. However, their current adoption in the industry seems to be marginal when compared to Java and others. Difficulties are related to the semantic intricacy of MTLs that despite their apparent simplicity (which helps introducing subtle critical errors); lack of debugging methods and tools; lack of performance, scalability, and inability to deal even with mid-sized models; little or no support for parallelisation, concurrent execution or distribution; and poor interoperability. In the same context, *bidirectionality in model transformations* is all-important as it permits two or more models to remain consistent while they undergo modifications. Current approaches often present idiosyncrasies that prevent the implementors from having complete control on the generated solutions. This is due to difficulties in assuring that a transformation is deterministic, making necessary in a class of problems the explicit management of the uncertainty related to the decision to pick the right solution. Understanding, which are the different application scenarios for deterministic and non-deterministic transformations, may mitigate the difficulties in adopting bidirectionality.

3.2 Domain Challenges

This dimension comprehends all the issues related to the nature of the application domains of the systems developed using MDE. Application domains like automotive, aerospace, nuclear, and healthcare aimed to assure a set of particular properties (i.e., privacy, security, safety). To reduce risks and to ensure that the software developed is reliable, *assurance case modeling* becomes an important part of the model-driven engineering techniques. At the same time, complex systems that also consider the social aspects (i.e., socio-technical systems), are composed of different and heterogeneous artifacts. A modeling framework to support the integration of data from sensors, open data, laws, regulations, scientific models (computational and data-intensive sciences), engineering models, and user preferences is needed (i.e., *DSLs for socio-technical integration*). Finally, the Internet of Things (IoT) domain represents a great opportunity for model-driven engineering applications in a wide range of domains, e.g., smart cities, smart buildings, industry 4.0, automotive, and health care. The answer that we still have to respond is: *Can MDE play a key role in the future of IoT and smart systems?*.

For sure, MDE allows coping with the complexity of reality by abstracting the relevant aspects for a particular application into corresponding models. In this respect, an MDE based solution is needed for smart city applications (i.e., in domains like Smart Mobility). MDE is a strategic piece of a framework to realize advanced solutions by taking into account different aspects and stakehold-

ers involved in the smart cities domain. Different views allow for the *separation of concerns* that, together to a higher level of abstraction, reduce the complexity of dealing with complex systems specification — continuous deployment and adaptation using MDE. The relationships between the views, their corresponding semantics, and the configuration of the different applications/services available in a city, constitute a megamodel [19]. In this dimension, in the last ten years, various engineering disciplines have emerged and are involved in the engineering process. How to transition from implicit to explicit knowledge about MDE in particular fields (i.e., Cyber-Physical Production Systems)?

3.3 Tool and implementation Challenges

Lack of good tooling is often mentioned as one key aspect hampering the adoption of MDE. We discussed that potential factors that may favour the adoption of model-driven development include adopting textual languages and treating the code as model; good and easy tooling (like modern IDEs); component-based solutions; and high-quality generated code.

Lots of interesting tools for building visual editors are currently available. Visualisation and visual editor frameworks are meant to help with working with complex problems, however too many difficulties are still encountered when designers use them for real. Thus, understanding the principles of building visual editors or visualisation frameworks that can apply to complex problems, and analysing where do our current frameworks/tools fall short should be a major concern.

Describing a complex system requires *modelling different heterogeneous views* [20] that need to be linked although they belong to different steps. Analysing how to build correspondences among such artefacts and understanding the semantics of such links is important in order to be able to insure traceability from requirements to implementation, and deduce requirements from the system. When several stakeholders are involved, artefacts must be linked to them as well and therefore understand the kind of requirements are existing. In this context, *expressing the requirements in a human-readable notation* that can be understood by a computer program can be highly relevant as well as and consequently understanding how to make the link among the involved artefact expressing the system and requirements in a same formalism (Single Model Principle).

Over the last decade, *scalability* has been denoted as one of the main challenges in model-driven engineering [17, 11]. As one participant pointed out, vanilla (out-of-the-box) EMF only works for simple projects. The problem is not just the size of the models but the diversity of artefacts, including models, metamodels, transformations, and dependencies, in any non-trivial project. There is a need to tame the accidental complexity of MDE itself. Running large transformations is as important as running transformations on large and heterogeneous models. Besides this, work on parallel and incremental querying and transformation is needed. In this sense, [21] defended the need for artifact models. According to [22] these artefacts should be viewed as data to which apply “classical” data analysis exploitation techniques (e.g., those coming from the information retrieval community).

Also, while participants agreed that we do have a reasonably robust MDE tool infrastructure (e.g., metamodeling and transformation languages), many core

MDE aspects could still be improved. [23] highlighted the need to simplify the creation of proper tool support for executable languages by providing various analysis tools for executable domain-specific modeling languages out-of-the-box based on single formalizations of their execution semantics. Similarly, [24] proposed a more general formalization of model synchronization and consistency management aspects that could be reused across different tools. This could also help with the challenge of making "chaining transformations" as straightforward as composing functions.

Another discussion point led to the argument that MDE tools should become more intelligent and self-aware. Several AI techniques could be used to cognify model-driven techniques [18] and to improve the autonomy of MDE tools (e.g., smart model autocompletion). Indeed, more and more MDE tools need to collaborate and agree on how to manage and evolve (runtime) models according to a shared set of goals [25]. Self-explanation capabilities will be critical in this scenario. This would also require considering *time* and timing issues as a first-class dimension (to be able to reason on when the model changes were done) as described in [26].

4 Social and Community Challenges

It emerged from the discussions that addressing the technical challenges often required also to consider social and community aspects in order to be able to validate such technical solutions or to be sure that it will be adopted in practice.

4.1 Social aspects

A critical discussion on social challenges took up the argument that MDE should be the catalyst to enable non-technical people to build the tools they need in their domains (Modelling by the People, for the People [27]). While this is one of the main selling arguments for MDE, it is still tricky for some modeling aspects, like the definition of desired consistency properties [28]. Ideally, instead of starting from scratch, stakeholders could be assisted in exploring the design space of potential models to be built [29], where these potential models (and their relationships) should be informed by domain information, e.g., regulatory texts from which some initial models could be inferred.

One possible solution, which was suggested during the discussion session, would be to facilitate deeper use of example-based modeling, even as a combination of formal and informal techniques to describe valid scenarios. It was also suggested to explore a kind of an Excel-like approach where one directly works at the instance-level all the time. It has also been proposed to expect less from the modelers, enabling practically useful analysis with minimal upfront modeling effort. Indeed, "how much modeling is enough" is a question that deserves to be explored, and that would help bridge modeling with agile approaches.

As a consequence of this discussion, the workshop considered whether any reluctance to employ MDE tools might be related to concerns over the Intellectual Property of the resulting models. This may be especially important in co-engineering projects where models are typically shared with third parties. Adapt-

ing well-known intellectual property management techniques (e.g., watermarking, fingerprinting, or obfuscation) to MDE artifacts may be one way forward to increasing confidence.

A more extreme suggestion involved moving to *domain-specific* MDE. Instead of considering MDE as if it was one general-purpose approach for systems and software engineering, we could start talking about "MDE for banking", 'MDE for insurance '", "MDE for health", and so on. Each domain might require different solutions, going far beyond the current approach of proposing different domain-specific languages for each sector. Domain-specific MDE could involve, for instance, tools explicitly tailored for different stakeholders in terms they understand (which may have a higher chance of being adopted).

4.2 Community aspects

Interestingly, it has been recognized from many sides how individuals can not easily address particular problems that instead affect the community as a whole and require more infrastructural solutions. One of the critical arguments made was that researchers and practitioners of MDE are primarily to blame for not having succeeded in selling the global software engineering community on the benefits of MDE. The workshop attendees challenged the notion of 'blame,' but acknowledged that the community would benefit from further MDE evangelism, as well as talking with software practitioners about MDE in a language that speaks to them.

There was also a strong consensus on the need for large model repositories where models could be endowed with confidence measures about their *quality*, e.g., via a community-based curation effort that tags the models contributed by others. This is especially needed for performing automated analysis that could influence the evolution of our field. However, quality assurance of models alone is not enough; we also need to ensure the representativeness of models (it was noted that most contributed models in existing repositories do not have constraints).

Another major community issue is how to teach students (who are the next generation of potential MDE practitioners and researchers). The workshop discussion considered whether we may need to change the way we teach MDE and focus first on teaching students on how to "use" MDE tools (and realizing the advantages of that) instead of teaching them how to "build" MDE tools. In the end, it is more likely that students end up belonging to the first group (MDE users) than to the second one (MDE builders) during their professional life. One way or the other, the workshop attendees concluded: setting up proper MDE teaching environments is still discouragingly hard.

Both richer model repositories and more MDE use-focused teaching require excellent collections of (reproducible, reusable, teachable) MDE projects, and not just individual models that anybody interested in MDE could easily import and explore [30].

5 Discussion

In this section, we briefly discuss the outcome of the proposed challenge classification. As aforementioned, the challenges have been arranged in different categories

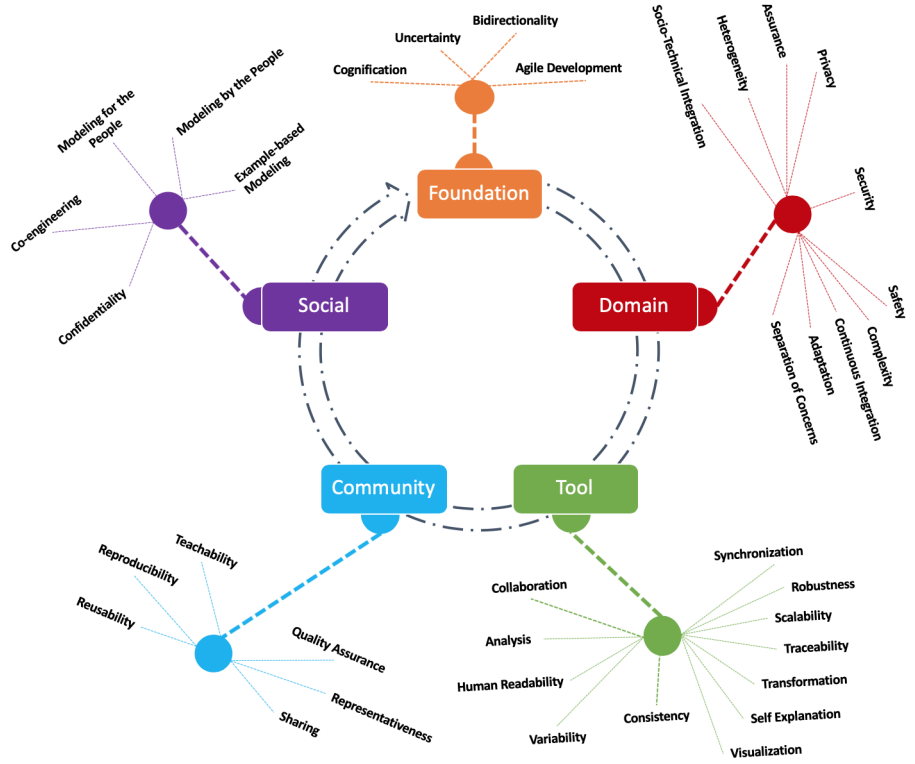

Fig. 1 Challenges Classification

that reflect the issues and problematic areas of the current state-of-art in model-driven engineering.

In particular, Fig. 1 illustrates such categories that, in turn, have been further refined to better characterize the challenge extension and boundary. Moreover, the main categories have been ordered according to their chronological relevance, e.g., the foundation challenges emerged before the domain and tool ones because most of the times tools have been developed for specific domains and based on theories and foundational elements. The advent of tools challenges somewhat corresponds also to an higher awareness about the limitations and difficulties in the practice of modeling partly due also to inflated expectations. For instance, the idea that most of the tools are lacking quality overwhelmingly emerged throughout the community that reacted in many different forms (e.g, publishing surveys on success stories and failures, organizing focused workshops and seminars, and so on). In other terms, the difficulties, which have been identified by the individual researchers and practitioners or within small organizations, started to be slowly part of a conventional wisdom. At the same time, social aspects become also relevant in many different directions, including collaborative modeling, confidentiality issues, and several forms of design-by-example.

6 Conclusions

In this article we presented the *grand challenges* in the Model-Driven Engineering field according to the expert participants in the two events we organized to discuss the future of MDE. We have classified them in different categories trying also to order them respect to their chronological relevance. We hope that this analysis not only represents a snapshot of the challenges faced in this research field but contributes to stimulate researchers, practitioners, and tool developers to tackle and explore some of them. At the same time, it provides a useful context for future research projects, research grant proposals and new research directions. We hope in a few years we can look back at this list and see many of them crossed out as a sign of the continuous advancement and maturity of our community.

References

1. Douglas C Schmidt. Model-driven engineering. *Computer-IEEE Computer Society*, 39(2):25, 2006.
2. Gunter Mussbacher, Daniel Amyot, Ruth Breu, Jean-Michel Bruel, Betty HC Cheng, Philippe Collet, Benoit Combemale, Robert B France, Rogardt Heldal, James Hill, et al. The relevance of model-driven engineering thirty years from now. In *International Conference on Model Driven Engineering Languages and Systems*, pages 183–200. Springer, 2014.
3. Ragnhild Van Der Straeten, Tom Mens, and Stefan Van Baelen. Challenges in model-driven software engineering. In *International Conference on Model Driven Engineering Languages and Systems*, pages 35–47. Springer, 2008.
4. Robert France and Bernhard Rumpe. Model-driven development of complex software: A research roadmap. In *2007 Future of Software Engineering*, pages 37–54. IEEE Computer Society, 2007.
5. Robert B. France and Bernhard Rumpe. The evolution of modeling research challenges. *Software and Systems Modeling*, 12(2):223–225, 2013.
6. Andy Evans, Robert B. France, Kevin Lano, and Bernhard Rumpe. The UML as a formal modeling notation. In *The Unified Modeling Language, UML'98: Beyond the Notation, First International Workshop, Mulhouse, France, June 3-4, 1998, Selected Papers*, pages 336–348, 1998.
7. Andy Evans and Stuart Kent. Core meta-modelling semantics of UML: the puml approach. In *UML'99: The Unified Modeling Language - Beyond the Standard, Second International Conference, Fort Collins, CO, USA, October 28-30, 1999, Proceedings*, pages 140–155, 1999.
8. Andy Evans, Robert B. France, Kevin Lano, and Bernhard Rumpe. Meta-modelling semantics of UML. In *Behavioral Specifications of Businesses and Systems*, pages 45–60. 1999.
9. Harald Störrle and Jan Hendrik Hausmann. Towards a formal semantics of UML 2.0 activities. In *Software Engineering 2005, Fachtagung des GI-Fachbereichs Softwaretechnik, 8.-11.3.2005 in Essen*, pages 117–128, 2005.
10. Jon Oldevik, Tor Neple, Roy Grønmo, Jan Øyvind Aagedal, and Arne-Jørgen Berre. Toward standardised model to text transformations. In *Model Driven Architecture - Foundations and Applications, First European Conference, ECMDA-FA 2005, Nuremberg, Germany, November 7-10, 2005, Proceedings*, pages 239–253, 2005.
11. Dimitrios S Kolovos, Louis M Rose, Nicholas Matragkas, Richard F Paige, Esther Guerra, Jesús Sánchez Cuadrado, Juan De Lara, István Ráth, Dániel Varró, Massimo Tisi, et al. A research roadmap towards achieving scalability in model driven engineering. In *Proceedings of the Workshop on Scalability in Model Driven Engineering*, page 2. ACM, 2013.
12. Ralf Laemmel. *Software Languages: Syntax, Semantics and Metaprogramming*. Springer-Verlag, 2018.
13. Gabriel A. Moreno and Paulo Merson. Model-driven performance analysis. In Steffen Becker, Frantisek Plasil, and Ralf Reussner, editors, *Quality of Software Architectures. Models and Architectures*. Springer, 2008.

14. Nelly Bencomo, Sebastian Götz, and Hui Song. Models@run.time: a guided tour of the state of the art and research challenges. *Software and Systems Modeling*, 18(5):3049–3082, 2019.
15. Robert France, Jim Bieman, and Betty HC Cheng. Repository for model driven development (ReMoDD). In *International Conference on Model Driven Engineering Languages and Systems*, pages 311–317. Springer, 2006.
16. Francesco Basciani, Juri Di Rocco, Davide Di Ruscio, Amleto Di Salle, Ludovico Iovino, and Alfonso Pierantonio. Mdeforge: an extensible web-based modeling platform. In *CloudMDE@ MoDELS*, pages 66–75, 2014.
17. Dimitrios S Kolovos, Richard F Paige, and Fiona AC Polack. The grand challenge of scalability for model driven engineering. In *International Conference on Model Driven Engineering Languages and Systems*, pages 48–53. Springer, 2008.
18. Jordi Cabot, Robert Clarisó, Marco Brambilla, and Sébastien Gérard. Cognifying model-driven software engineering. In Seidl and Zschaler [31], pages 154–160.
19. Jean-Marie Favre and Tam Nguyen. Towards a megamodel to model software evolution through transformations. *Electron. Notes Theor. Comput. Sci.*, 127(3):59–74, April 2005.
20. Hugo Brunelière, Erik Burger, Jordi Cabot, and Manuel Wimmer. A feature-based survey of model view approaches. *Software and Systems Modeling*, 18(3):1931–1952, 2019.
21. Arvid Butting, Timo Greifenberg, Bernhard Rumpe, and Andreas Wortmann. On the need for artifact models in model-driven systems engineering projects. In Seidl and Zschaler [31], pages 146–153.
22. Önder Babur, Loek Cleophas, Mark van den Brand, Bedir Tekinerdogan, and Mehmet Aksit. Models, more models, and then a lot more. In Seidl and Zschaler [31], pages 129–135.
23. Tanja Mayerhofer and Benoît Combemale. The tool generation challenge for executable domain-specific modeling languages. In Seidl and Zschaler [31], pages 193–199.
24. Zinovy Diskin, Harald König, Mark Lawford, and Tom Maibaum. Toward product lines of mathematical models for software model management. In Seidl and Zschaler [31], pages 200–216.
25. Antonio García-Domínguez and Nelly Bencomo. Non-human modelers: Challenges and roadmap for reusable self-explanation. In Seidl and Zschaler [31], pages 161–171.
26. Robert Bill, Alexandra Mazak, Manuel Wimmer, and Birgit Vogel-Heuser. On the need for temporal model repositories. In Seidl and Zschaler [31], pages 136–145.
27. Steven Kelly. Modelling by the people, for the people. In Seidl and Zschaler [31], pages 178–183.
28. Martin Gogolla, Frank Hilken, and Andreas Kästner. Some narrow and broad challenges in MDD. In Seidl and Zschaler [31], pages 172–177.
29. Vinay Kulkarni and Sreedhar Reddy. From building systems right to building right systems - A generic architecture and its model based realization. In Seidl and Zschaler [31], pages 184–192.
30. Juri Di Rocco, Davide Di Ruscio, Ludovico Iovino, Ralf Laemmel, and Alfonso Pierantonio. MDE adoption — a three-legged chair. In *Proc. Workshop on Grand Challenges in Modeling at STAF*, 2017.
31. Martina Seidl and Steffen Zschaler, editors. *Software Technologies: Applications and Foundations - STAF 2017 Collocated Workshops, Marburg, Germany, July 17-21, 2017, Revised Selected Papers*, volume 10748 of *Lecture Notes in Computer Science*. Springer, 2018.