MONDO: Scalable Modelling and Model Management on the Cloud

Dimitrios S. Kolovos[1], Louis M. Rose[1], Richard F. Paige[1], Esther Guerra[2], Jesús Sánchez Cuadrado[2], Juan De Lara[2], István Ráth[3], Dániel Varró[3], Gerson Sunyé[4], Massimo Tisi[4]
{dimitris.kolovos, louis.rose, richard.paige}@york.ac.uk, {Esther.Guerra, Jesus.Sanchez.Cuadrado, Juan.deLara}@uam.es, {rath, varro}@mit.bme.hu, {gerson.sunye, massimo.tisi}@inria.fr

[1]University of York,
[2]Universidad Autónoma de Madrid,
[3]Budapest University of Technology and Economics,
[4]AtlanMod (Inria, Mines Nantes, LINA)

Abstract. Achieving scalability in modelling and MDE involves being able to construct large models and domain-specific languages in a systematic manner, enabling teams of modellers to construct and refine large models in collaboration, advancing the state of the art in model querying and transformations tools so that they can cope with large models (of the scale of millions of model elements), and providing an infrastructure for efficient storage, indexing and retrieval of large models. This paper outlines how MONDO, a collaborative EC-funded project, contributes to tackling some of these scalability-related challenges.

1 Project Identity

- **Project acronym:** MONDO
- **Project title:** Scalable Modelling and Model Management on the Cloud
- **Project partners:** The Open Group (Project Coordinator), University of York (Technical Coordinator), Autonomous University of Madrid, University of Nantes, Budapest University of Technology and Economics, IKERLAN, SOFTEAM, Soft-Maint, UNINOVA
- **Website:** `www.mondo-project.org`
- **Project start date/duration:** Nov 1, 2013 (30 months)

2 Introduction

As MDE is increasingly applied to larger and more complex systems, the current generation of modelling technologies are being stressed to their limits in terms of their capacity to accommodate collaborative development, efficient management and persistence of models larger than a few hundreds of megabytes in size. In our view, achieving scalability in MDE involves:

- being able to construct large models and domain specific languages in a systematic manner;
- enabling large teams of modellers to construct and refine large models in a collaborative manner;
- advancing the state of the art in model querying and transformations tools so that they can cope with large models (with millions of model elements);
- providing an infrastructure for efficient storage, indexing and retrieval of such models.

The rest of the paper (Sections 3-6) provides an overview of the state of the art in these four key areas, identifies the main challenges that need to be overcome, and outlines the realised and envisioned contributions of MONDO. Section 7 concludes the paper.

3 Scalable Domain Specific Languages

In current MDE practice, we still find that domain specific modelling languages (DSMLs) are often constructed in an ad-hoc way. Moreover, graphical DSMLs scale poorly for large models. There are several works aimed at defining compositional mechanisms for languages and models. [1] provides composition techniques for languages lacking such built-in mechanisms. Other works extend meta-models [2] and models [3] with export and import interfaces, but are only described theoretically.

With respect to visualizing large models, some researchers have brought techniques from the field of information visualization, like semantic zooming [4]. Some language editors DiaGen enable the definition of abstractions [5], but they have to be manually programmed for each different language. Reusable model abstractions are reported in [6], but with no support for visualizations.

Finally, little work has addressed processes for developing and testing meta-models [7, 8]. In [8] a language to write automated tests for conceptual schemas is proposed, while [7] proposes the incremental development of meta-models by increasingly refined test models. Other works have explored the induction of meta-models from example models [9, 10]. However, none of these works consider issues related to scalability of models or meta-models.

3.1 Research Directions

Scalable Language Design MONDO addresses both the engineering of large DSMLs, and DSMLs expecting large models. For the first issue, we make available reusable patterns, accounting for the several aspects in the definition of a DSML, including abstract, concrete syntax, semantics, and the services of the modelling environment like model fragmentation, filtering and conformance checking. The use of patterns for all these aspects facilitate and speed up DSML definition. We have created a tool, called `DSL-tao` (`http://jdelara.github.io/DSL-tao/`), which permits describing DSMLs and their environments using this

philosophy. Second, we propose fragmenting models following modular principles adopted by many programming languages [11]. Therefore a model is organized as a Project, and then be fragmented into Packages (which are mapped to folders in the file system), which may hold Units (mapped in files). This fragmentation strategy is specified at the meta-model level, by instantiating a pattern.

Scalable Visualization Another means to tackle scalability is to support useful abstractions, providing a simplified view of a model, or introducing hierarchical elements, organizing models at different levels of abstraction. In our approach, these abstractions are specified as patterns. Another common issue is that no concrete syntax (only generic tree-based editors) is defined for some meta-models, which becomes problematic as models grow. To address such scenarios, we have created a tool called SAMPLER (http://rioukay.github.io/sampler/), which allows the scalable visual exploration and navigation of EMF models.

Processes and Methodologies Currently, DSMLs are often developed in an informal, ad-hoc way. However, DSMLs should be engineered using sound principles and methods, gathering requirements from all stakeholders. We are currently developing methodologies [12] including validation and verification mechanisms [13, 14] (based on DSLs to specify different kinds of tests) for engineering DSMLs, which are being integrated with the previous tools.

4 Scalable Queries and Transformations

Some work exists that applies either incrementality, laziness or distribution to MDE. *Incremental computation* has been used for transforming large evolving models, either with live [15] (i.e., transformation during the update) or offline [16] (i.e., transformation after the update) incrementality. Incremental graph transformation approaches [17–19] focus especially on techniques for incremental pattern-matching. *Lazy computation* can improve scalability when only a small part of large models is accessed. A model transformation tool with an lazy/on-demand generation strategy has been proposed in [20]. The Stratego [21] system allows user-defined execution strategies for transformation rules, that may be in principle used for on-demand transformation. VIATRA lazily evaluates the matches of connected rules to avoid unnecessary computation [22]. Lazy loading [23] allows to handle models that do not fit into the available memory. *Distributed computation* is convenient for complex and parallelizable transformations. In graph transformation, recent work [24] focuses on parallelizing the recursive matching phase, particularly expensive for graph transformations. In model transformation, Lintra [25] allows to specify distributed transformations by explicit distribution primitives.

4.1 Research Directions

Benchmarks for Scalable Query and Transformation We defined a set of shared benchmarks for query and transformations on very large models, gathering real-

world MDE case studies. We have made these cases available to the community for evaluation via a public MDE benchmark repository [26].

Reactive Model Query and Transformation We propose a shift of paradigm for programming MDE applications towards reactive programming [27], where a network of transformations defines persistent data-flows among models. A reactive transformation engine takes care of activating only the strictly needed computation in response to updates or requests of model elements, by a combination of incremental and lazy computation. A reactive engine also opens the way to scenarios based on infinite intermediate models generated on demand, or streaming models propagating from inputs to outputs.

Distributed Model Query and Transformation in the Cloud We propose engines that implicitly (i.e., without explicit distribution primitives) distribute the execution of model queries and transformations on top of well-known distributed programming models for the Cloud (e.g., MapReduce). We show how the execution semantics of model queries and transformations can be aligned with parallel computation models [28].

Prototypes for several components are already available: IncQuery-D, a scalable engine for distributed incremental model queries [29]; MONDO-SAM, a benchmarking framework [30]; VIATRA-CEP, a streaming transformation engine [31].

5 Scalable Collaborative Modelling

Model repositories such as CDO [32] and MORSA [33] are storage systems for modeling artefacts that are mostly focused on concurrent access over a client-server infrastructure. They provide extension mechanisms and core APIs that auxiliary, function-specific tools may use to support conflict management, branching, model comparison etc.

Online collaborative modelling systems such as CoolModes [34] rely on a short transaction approach, whereby a single, shared instance of the model is edited by multiple users concurrently. These systems lack conflict management, or only provide very light weight mechanisms (such as voluntary locking).

Model versioning systems such as EMFStore [35] are more closely aligned with offline version control systems such as SVN. They follow the *long transaction approach* whereby contributors are assumed to commit larger portions of work with respect to a certain (past) version as the reference. Hence, since conflicts are common, their detection, resolution and merging are features of top importance. To that end, such sytems are frequently augmented with *offline model comparison, differencing and merging tools* such as EMF Compare [36] or EMF Diff/Merge [37] are also often used.

The key weaknesses of the collaborative modelling state of the art can be summarized as follows: (i) immature integration of online and offline collaboration patterns; (ii) mostly ad-hoc architectures that prohibit or make the implementation of domain-specific collaboration/version management difficult; (iii) very

simplistic locking and conflict management solutions that severely hinder developer productivity; (iv) the lack of a flexible *and* scalable back-end platform that caters to both Eclipse-based and other (commercial) tools.

5.1 Research Directions

We are working on a multi-device collaborative modelling framework which on the front-end is fully compatible with existing and future Eclipse-based technologies (EMF and its auxiliaries and the Team API); on the back-end, and integrates into the scalable model persistence framework. It *supports both offline and online collaboration* in a multi-user and multi-device environment, providing a model access layer (transaction management, queries, views and manipulation) featuring basic collaboration primitives (push, pull, commit, merge), and an adaptation layer for the integration of access control and security services. It is built on an extensible architecture that allows the integration of domain-specific, customized plugins for conflict management (detection, resolution and merging).

As novel and innovative features, it includes *query-driven dynamic locking* that uses complex graph queries [38] for the specification of locking partitions for views and manipulative transactions. Such queries operate in a collaboration-aware manner that includes support for real-time updates and locked queries (where updates are propagated only from a pre-defined subset of collaboration partners). Additionally, the framework features *automated conflict resolution* based on design-space exploration techniques [39] that are able to ensure domain consistency and well-formedness by automatically applying model manipulation policies to find valid and conflict-free model states. Prototypes for both the collaboration framework and the model merger based on design space exploration are already available under `http://github.com/FTSRG/mondo-collab-framework` and `http://github.com/FTSRG/mondo-collab-mergespaceexploration`.

6 Scalable Model Persistence

An essential component of scalable MDE is infrastructure that facilitates persistence and retrieval or large models in an efficient manner.

Efficient Model Storage The current standard model storage format is the XML Metadata Interchange. As XMI is an XML-based format, in order to access any model elements using current state-of-the-art modelling frameworks such as EMF, the complete model file needs to be parsed and loaded in memory first. This implies that the larger the model file, the more time and memory is needed in order to load the model. Also, XMI inherits the verbosity of XML which means that XMI-encoded model files are much larger in size than needed in order to store the information they do.

To address these issues, we envision a new efficient model representation format that will reduce the size of model files, enable modelling and model

management tools to lazily load the contents of a model into memory, and access specific model elements without needing to read the entire model file first. We anticipate that such a format will provide a substantial improvement both in terms of both the size of model files, and in terms of the memory and time required to load these models.

Model Indexing With a faster and more efficient model persistence format that provides a reduction of the scale of 10 in terms of size, an XMI-based model of the order of hundreds of MBs, would now be of the order of tens of MBs. In a typical collaborative development environment where artefacts are stored in a central repository (e.g. CVS, SVN, Git etc.), even files of the order of tens of MBs are challenging to manage as for every change they need to be transferred back and forth between the local copy and the remote repository. Storing a large model as a single file can also be sub-optimal as it can cause frequent conflicts when using an optimistic locking VCS or lock-outs when using a pessimistic locking VCS. Two solutions have been proposed for addressing this problem [40]: 1) Storing large models in dedicated model repositories that enable model-element level (instead of file-level) version control operations (check in, check out, lock etc.), and 2) Splitting large models over many cross-referenced physical files (model fragments).

The first approach requires both a leap in terms of the modelling tools used to edit models, as the majority of modelling tools work with file-based models, and a transition from robust and established repositories which work well with a wide range of development tools, to newly developed model-specific repositories. The particularly limited adoption of model-specific repositories such as CDO, and ModelCVS [41] so far has demonstrated that industrial users can be reluctant to make such a drastic transition in practice. As such, and in order to provide industrially-relevant results, we will mainly focus on the second approach. The main advantage of the second approach is that it works well with

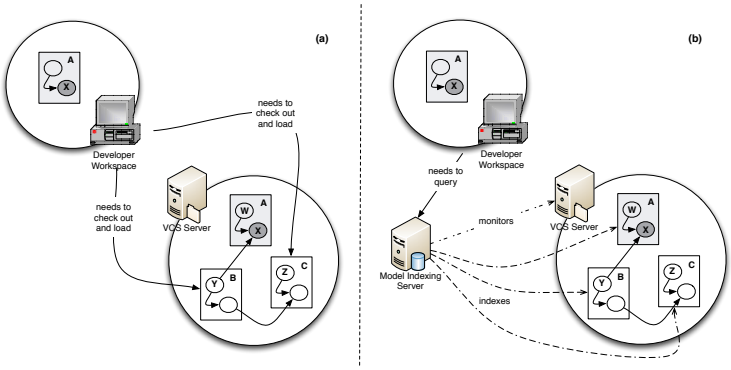

Fig. 1. Performing global queries on model fragments stored in a VCS repository without (a) and with (b) an indexing server

existing modelling tools, and with existing types of remote repositories (such

as CVS, SVN, Git, FTP, shared network folders etc). However, using this approach with current state-of-the-art technologies makes it impossible to compute queries of global nature without going through all the model fragments from the remote repository every time. For example, consider the scenario on the left side (a) of Figure 1, where the VCS repository contains 3 model fragments (A, B and C) from which the developer has checked out only fragment A. Now, if the developer needs to know which other fragments in the repository reference its X element, they need to check out, load and examine every other fragment in the repository (B and C in this case). Obviously, as the number of model fragments in the repository grows, this approach becomes increasingly inefficient.

To address this limitation, we are working on a model indexing framework (Hawk [42] - `https://github.com/kb634/mondo-hawk`) that can monitor the contents of remote version control repositories, and index the models they contain in a scalable database that will enable efficient computation of global queries. Hawk can support monitoring different types of remote repositories (SVN, Git etc.) and indexing of heterogeneous models (i.e. XMI, proprietary) using a driver-based architecture.

7 Conclusions and Next Steps

In this paper we have provided an outline of the main scalability challenges in MDE, and MONDO's technical vision for addressing them. MONDO has already contributed novel techniques and several prototype implementations in all four identified key-areas. During the last year of the project, we plan to increasingly focus on integrating these prototypes in the context of a unified technical offering in preparation for the evaluation phase of the project, where the research contributions of MONDO will be assessed in the context of four industrial case studies.

The first case study (provided by UNINOVA[1]) comes from the construction industry and involves collaborative development and automated management of large computer-aided design (CAD) models. The second case study (provided by Soft-Maint[2]) involves exploration and automated transformation of large models which have been reverse-engineered from existing codebases. The third case study (provided by IKERLAN[3]) involves multi-device collaborative development of models from the offshore wind power industry, and the fourth case study involves managing large collections of UML models stored in a proprietary format supported by Softeam's[4] Modelio[5] tool.

[1] `http://www.uninova.pt/`
[2] `http://www.sodifrance.fr/`
[3] `http://www.ikerlan.es/`
[4] `http://www.softeam.fr/`
[5] `https://www.modeliosoft.com/`

References

1. Florian Heidenreich, Jakob Henriksson, Jendrik Johannes, and Steffen Zschaler. On language-independent model modularisation. *T. Aspect-Oriented Software Development VI*, 6:39–82, 2009.
2. Ingo Weisemöller and Andy Schürr. Formal definition of MOF 2.0 metamodel components and composition. In *Model Driven Engineering Languages and Systems, 11th International Conference*, volume 5301 of *Lecture Notes in Computer Science*, pages 386–400. Springer, 2008.
3. Stefan Jurack and Gabriele Taentzer. A component concept for typed graphs with inheritance and containment structures. In *Graph Transformations - 5th International Conference, ICGT 2010*, volume 6372 of *Lecture Notes in Computer Science*, pages 187–202. Springer, 2010.
4. Mathias Frisch, Raimund Dachselt, and Tobias Brückmann. Towards seamless semantic zooming techniques for UML diagrams. In *SOFTVIS*, pages 207–208. ACM, 2008.
5. Oliver Köth and Mark Minas. Structure, abstraction, and direct manipulation in diagram editors. In *Diagrams*, volume 2317 of *LNCS*, pages 290–304. Springer, 2002.
6. Juan de Lara, Esther Guerra, and Jesús Sánchez Cuadrado. Abstracting modelling languages: A reutilization approach. In *Proc. CAiSE 2012*, volume 7328 of *Lecture Notes in Computer Science*, pages 127–143. Springer, 2012.
7. Antonio Cicchetti, Davide Di Ruscio, Dimitris Kolovos, and Alfonso Pierantonio. A test-driven approach for metamodel development. In *Emerging Technologies for the Evolution and Maintenance of Software Models*, pages 319–342. IGI Global, 2012.
8. Albert Tort and Antoni Olivé. An approach to testing conceptual schemas. *Data Knowl. Eng.*, 69(6):598–618, 2010.
9. Hyun Cho, Yu Sun, Jeff Gray, and Jules White. Key challenges for modeling language creation by demonstration. In *ICSE'11 Workshop on Flexible Modeling Tools*, 2011.
10. Jesús Sánchez Cuadrado, Juan de Lara, and Esther Guerra. Bottom-up meta-modelling: An interactive approach. In *Model Driven Engineering Languages and Systems - 15th International Conference, MODELS 2012, Innsbruck, Austria, September 30-October 5, 2012. Proceedings*, volume 7590 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2012.
11. Antonio Garmendia, Esther Guerra, Dimitrios S. Kolovos, and Juan de Lara. EMF splitter: A structured approach to EMF modularity. In *XM@MoDELS*, volume 1239 of *CEUR Workshop Proceedings*, pages 22–31. CEUR-WS.org, 2014.
12. Jesús Juan López-Fernández, Jesús Sánchez Cuadrado, Esther Guerra, and Juan de Lara. Example-driven meta-model development. *SoSyM*, in press, 2014, see also `http://www.miso.es/tools/metaBUP.html`.
13. Jesús J. López-Fernández, Esther Guerra, and Juan de Lara. Assessing the quality of meta-models. In *MoDeVVa@MODELS*, volume 1235 of *CEUR Workshop Proceedings*, pages 3–12. CEUR-WS.org, 2014.
14. Jesús J. López-Fernández, Esther Guerra, and Juan de Lara. Meta-model validation and verification with metabest. In *ACM/IEEE ASE*, pages 831–834. ACM, 2014.
15. Frédéric Jouault and Massimo Tisi. Towards incremental execution of ATL transformations. In *Theory and Practice of Model Transformations*, pages 123–137. Springer, 2010.

16. Yingfei Xiong, Dongxi Liu, Zhenjiang Hu, Haiyan Zhao, Masato Takeichi, and Hong Mei. Towards automatic model synchronization from model transformations. *Proc. of ASE'07*, page 164, 2007.
17. G. Bergmann, I. Ráth, and D. Varró. Parallelization of graph transformation based on incremental pattern matching. *Electronic Communications of EASST*, 18, 2009.
18. Gabor Bergmann, Istvan Rath, Gergely Varro, and Daniel Varro. Change-driven model transformations. *Software and Systems Modeling*, pages 1–31, 2011. 10.1007/s10270-011-0197-9.
19. Holger Giese and Robert Wagner. From model transformation to incremental bidirectional model synchronization. *Software & Systems Modeling*, 8(1):21–43, 2008.
20. Massimo Tisi, Salvador Martínez, Frédéric Jouault, and Jordi Cabot. Lazy execution of model-to-model transformations. In *Model Driven Engineering Languages and Systems*, pages 32–46. Springer, 2011.
21. Eelco Visser. Program transformation with Stratego/XT: Rules, strategies, tools, and systems in Stratego/XT 0.9. In *Domain-Specific Program Generation*, volume 3016 of *LNCS*, pages 216–238. Springer, 2003.
22. Gabriele Taentzer, Karsten Ehrig, Esther Guerra, J. de Lara, L. Lengyel, Tihamer Levendovszky, Ulrike Prange, D. Varró, and S. Varró-Gyapay. Model transformation by graph transformation: A comparative study. In *Proc. Workshop Model Transformation in Practice*, 2005.
23. Jouault, Frédéric and Bézivin, Jean and Barbero, Mikaël. Towards an advanced model-driven engineering toolbox. *Innovations in Systems and Software Engineering*, 5:5–12, 2009.
24. Gergely Mezei, Tihamer Levendovszky, Tamas Meszaros, and Istvan Madari. Towards Truly Parallel Model Transformations: A Distributed Pattern Matching Approach. In *IEEE EUROCON 2009*, pages 403–410. IEEE, 2009.
25. Loli Burgueño, Javier Troya, Manuel Wimmer, and Antonio Vallecillo. On the Concurrent Execution of Model Transformations with Linda. In *Proceeding of the First Workshop on Scalability in MDE*, BigMDE '13, pages 3:1–3:10, New York, NY, USA, 2013. ACM.
26. Amine Benelallam, Massimo Tisi, Istvan Rath, Benedeck Izso, and Dimitrios S. Kolovos. Towards an Open Set of Real-World Benchmarks for Model Queries and Transformations. In CEUR Workshop Proceedings, editor, *BigMDE*, York, UK, United Kingdom, July 2014. University of York.
27. D. Harel and A. Pnueli. *On the development of reactive systems*, pages 477–498. Springer-Verlag New York, Inc., New York, NY, USA, 1985.
28. Massimo Tisi, Frdric Jouault, Jrme Delatour, Zied Saidi, and Hassene Choura. fuml as an assembly language for model transformation. In Benot Combemale, DavidJ. Pearce, Olivier Barais, and JurgenJ. Vinju, editors, *Software Language Engineering*, volume 8706 of *Lecture Notes in Computer Science*, pages 171–190. Springer International Publishing, 2014.
29. Gábor Szárnyas, Benedek Izsó, István Ráth, Dénes Harmath, Gábor Bergmann, and Dániel Varró. Incquery-d: A distributed incremental model query framework in the cloud. In *ACM/IEEE 17th International Conference on Model Driven Engineering Languages and Systems, MODELS 2014*, Valencia, Spain, 2014. Springer, Springer. Acceptance rate: 26%.
30. Benedek Izsó, Gábor Szárnyas, István Ráth, and Dániel Varró. Mondo-sam: A framework to systematically assess mde scalability. In *BigMDE 2014 2nd Workshop on Scalable Model Driven Engineering*, page 40. ACM, ACM, 2014.

31. István Dávid, István Ráth, and Dániel Varró. Streaming model transformations by complex event processing. In Juergen Dingel and Wolfram Schulte, editors, *ACM/IEEE 17th International Conference on Model Driven Engineering Languages and Systems, MODELS 2014*, Valencia, Spain, 2014. Springer, Springer. Acceptance rate: 26%.
32. Eclipse. The connected data objects model repository (CDO) project, 2012. `http://eclipse.org/cdo`.
33. Javier Espinazo Pagan, Jesus Sanchez Cuadrado, and Jesus García Molina. Morsa: A scalable approach for persisting and accessing large models. In Jon Whittle, Tony Clark, and Thomas Kühne, editors, *Model Driven Engineering Languages and Systems*, volume 6981 of *Lecture Notes in Computer Science*, pages 77–92. Springer Berlin / Heidelberg, 2011. 10.1007/978-3-642-24485-8_7.
34. Niels Pinkwart. A Plug-In Architecture for Graph Based Collaborative Modeling Systems. In V. Aleven et al, editor, *Supplementary Proceedings of the 11th Conference on Artificial Intelligence in Education, Sydney (Australia)*, pages 89–94, Sydney, Australia, 2003. SIT.
35. Maximilian Koegel and Jonas Helming. Emfstore: a model repository for emf models. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2*, ICSE '10, pages 307–308, New York, NY, USA, 2010. ACM.
36. C. Brun et al. EMF compare 2.0, 2012. `http://www.eclipse.org/emf/compare/`.
37. O. Constant. EMF Diff/Merge, 2012. `http://eclipse.org/diffmerge/`.
38. Zoltán Ujhelyi, Gábor Bergmann, Ábel Hegedüs, Ákos Horváth, Benedek Izsó, István Ráth, Zoltán Szatmári, and Dániel Varró. Emf-incquery: An integrated development environment for live model queries. *Science of Computer Programming*, 98, 02/2015 2015.
39. Ábel Hegedüs, Ákos Horváth, and Dániel Varró. A model-driven framework for guided design space exploration. *Automated Software Engineering*, pages 1–38, 08/2014 2014.
40. Dimitris Kolovos and Richard Paige and Fiona Polack. The Grand Challenge of Scalability for Model Driven Engineering. In Chaudron, Michel, editor, *Models in Software Engineering*, volume 5421 of *Lecture Notes in Computer Science*, pages 48–53. Springer Berlin / Heidelberg, 2009.
41. G. Kramler, G. Kappel, T. Reiter, E. Kapsammer, W. Retschitzegger, and W. Schwinger. Towards a semantic infrastructure supporting model-based tool integration. In *Proceedings of the 2006 international workshop on Global integrated model management*, GaMMa '06, pages 43–46, New York, NY, USA, 2006. ACM.
42. Konstantinos Barmpis and Dimitrios S. Kolovos. Towards scalable querying of large-scale models. In Jordi Cabot and Julia Rubin, editors, *Modelling Foundations and Applications*, volume 8569 of *Lecture Notes in Computer Science*, pages 35–50. Springer International Publishing, 2014.