Models, More Models, and Then a Lot More

Önder Babur[1(✉)], Loek Cleophas[1], Mark van den Brand[1], Bedir Tekinerdogan[2], and Mehmet Aksit[3]

[1] Eindhoven University of Technology, Eindhoven, The Netherlands
{O.Babur,L.G.W.A.Cleophas,M.G.J.v.d.Brand}@tue.nl
[2] Wageningen University & Research, Wageningen, The Netherlands
Bedir.Tekinerdogan@wur.nl
[3] University of Twente, Enschede, The Netherlands
M.Aksit@utwente.nl

Abstract. With increased adoption of Model-Driven Engineering, the number of related artefacts in use, such as models, metamodels and transformations, greatly increases. To confirm this, we present quantitative evidence from both academia — in terms of repositories and datasets — and industry — in terms of large domain-specific language ecosystems. To be able to tackle this dimension of scalability in MDE, we propose to treat the artefacts as data, and apply various techniques — ranging from information retrieval to machine learning — to analyse and manage those artefacts in a holistic, scalable and efficient way.

Keywords: Model-Driven Engineering · Scalability
Model analytics · Data mining · Machine learning

1 Introduction

Model-Driven Engineering (MDE) promotes the use of models, metamodels and model transformations as first-class citizens to tackle the complexity of software systems. As MDE is applied to larger problems, the complexity, size and variety of those artefacts increase. With respect to model size and complexity, for instance, the aspect of scalability has been pointed out by Kolovos et al. [18]. Regarding this aspect, a good amount of research has been done for handling a small number of (possibly very big and complex) models, e.g. in terms of comparison, merging, splitting, persistence or transformation. However, scalability with respect to model variety and multiplicity (i.e. dealing with a large number of possibly heterogeneous models) has so far remained mostly under the radar.

In this paper, we advocate this aspect of scalability as a potentially big challenge for broader MDE adoption. We highlight evidence and concerns which cross-cut the dichotomies of industry vs. academia and of open source vs. commercial software. We thus show that scalability proves to be an issue overall. Furthermore, we mention several related domains and disciplines as inspiration for tackling scalability, with pointers to some related work. Yet we note the

© Springer International Publishing AG 2018
M. Seidl and S. Zschaler (Eds.): STAF 2017 Workshops, LNCS 10748, pp. 129–135, 2018.
https://doi.org/10.1007/978-3-319-74730-9_10

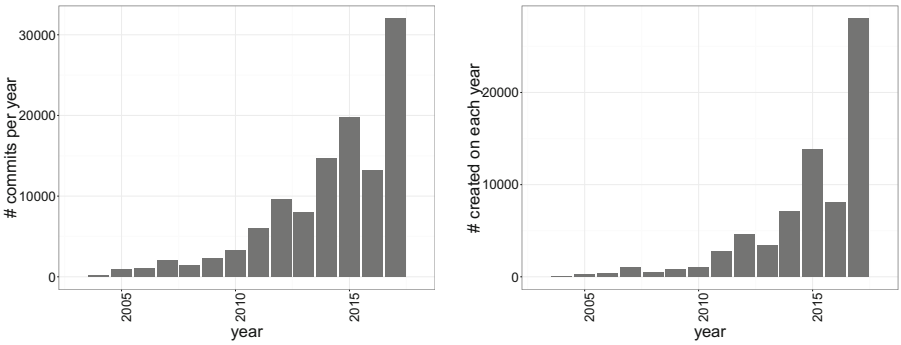
inherent differences of MDE artefacts (models in particular), compared to common types of data such as natural language text and source code. This might render it difficult to directly apply techniques from other domains to MDE.

2 The Expanding Universe of MDE

The aforementioned scalability issue emerges partly due to some recent developments in the MDE community. Firstly, there have been efforts to initiate public repositories to store and manage large numbers of models and related artefacts [5,23]. Further efforts include mining public repositories for MDE-related items from GitHub, e.g. Eclipse-based MDE technologies [17] and UML models [14] (the Lindholmen dataset). In the latter, the number of UML models can go up to more than 90k. The sheer number of models inevitably calls for techniques for searching, preprocessing (e.g. filtering), analysing and visualising the data in a holistic and efficient manner.

Mini Study: Ecore Metamodels in GitHub. Kolovos et al. present a study on the use of MDE technologies in GitHub [17]. Among a rich set of empirical results, they report the number of search results for Ecore metamodels in GitHub (as of early 2015) to be ∼15k, and show a rapidly increasing trend in the number of commits on MDE-related files. We were triggered by the fact that the same exercise of searching Ecore files, with the query reported in the paper (https://github.com/search?q=EClass+extension:ecore&type=Code) yields (as of September 2017) more than 67k results; a fourfold increase. Here we demonstrate a mini case study on the Ecore metamodel files in GitHub over time. We slightly relaxed the query by replacing the search term *EClass* with *ecore*, and mined all the files together with the relevant commits on them. Figure 1 depicts a strong upward trend for (a) the number of commits per year on Ecore files, and (b) the number of newly created Ecore files per year. The sharp increase in 2017 can be partly attributed to recently emerging metamodel repositories and datasets on GitHub; nevertheless as the end effect there are increasingly more Ecore metamodels in GitHub.

MDE in the Industry. Even within a single industry or organisation, a similar situation emerges with increased adoption of MDE. We have been collaborating with high tech companies in the Netherlands. One of those companies maintains a set of MDE-based domain-specific language (DSL) ecosystems. Just a single one of those ecosystems currently contains dozens of metamodels, thousands of models and tens of thousands LOC of transformations. With the complete revision history, the total number of artefacts goes up to tens of thousands. Another company, which applies MDE in six different projects, reports a similar collection of thousands of artefacts based on various technologies (e.g. different transformation languages). Similar stories in terms of scale hold for our other industrial partners, with growing heterogeneous sets of artefacts involving multiple domains. Note that for systems with implicit or explicit (e.g. as a Software Product Line) variability, *variants* can be considered another amplifying factor besides *versions* for the total number of MDE artefacts to manage.



(a) Number of commits per year (b) New files per year

Fig. 1. GitHub results on Ecore metamodels.

Along with conventional forward engineering approaches, we observe an increasing trend in our partners with legacy software: automated migration into model-driven/-based engineering using process mining and model learning. All the presented facts let us confirm the statement by Brambilla et al. [8] and Whittle et al. [24] that MDE adoption in (at least some parts of) the industry grows quite rapidly, and we conclude that tackling scalability will be increasingly important in the future.

3 Treating MDE Artefacts as Data

Based on the observations above, we advocate a perspective where MDE artefacts are treated holistically as data, processed and analysed with various scalable and efficient techniques, possibly inspired by related domains. Tackling large volumes of artefacts has been commonplace in other domains, such as text mining for natural language text [15], and repository mining for source code [16]. While we might not be able to apply those techniques as-is on MDE-related artefacts, the general workflow remains as a rough guideline with several steps of data collection, cleaning, integration and transformation, feature engineering and selection, modelling (e.g. as statistical models, neural networks), and finally deployment, exploration and visualisation.

To exemplify the different nature of MDE data and hence the different requirements on its analysis, take the problem of clone detection. Clone detection on source code is already several steps away from text mining, as code clone detection usually involves steps such as a language-specific parsing of the code into abstract syntax trees (AST), normalisation of identifiers, and structural transformations [21]. Model clone detection, on the other hand, possess further challenges. To cite Deissenboeck et al., *"Algorithms from code clone detection are only of minor interest for model clone detection, as they usually work on either a linear text or token stream or on the tree structured AST of the code, which both are not transferable to general directed graphs."* [11]. Furthermore,

Störrle points out inherent differences of models compared to code, including CASE tool integration and tool-specific representations, internal identifiers and different layouts with no semantic implications, abstract vs. (possibly multiple) concrete syntaxes, etc. [22]. The case of clone detection reinforces our argument that techniques from related domains such as data mining and repository mining might not be directly translatable to the MDE domain.

4 Relevant Domains for Model Analytics

Despite the different nature of models as exemplified above, we can get inspired by the techniques from other disciplines and try to adapt them for the problems in MDE. As a preliminary overview, in this section we list and discuss several such domains. While there is related MDE research on some of the items on the list, we believe a conscious and integrated mindset would mitigate the challenges for scalable MDE.

Descriptive Statistics. Several MDE researchers have already performed empirical studies on MDE artefacts with a statistical mindset. For instance, Kolovos et al. assesses the use of Eclipse technologies in GitHub, giving related trend analyses [17]. Mengerink et al. present an automated analysis framework on version control systems with similar capabilities [19]. Di Rocco et al. perform a correlation analysis on metrics for various MDE artefacts [12]. Descriptive statistics could in the most general sense be exploited to gain insights over large numbers of MDE arfefacts in terms of general characteristics, patterns, outliers, statistical distributions, dependence, etc.

Information Retrieval. Techniques from information retrieval (IR) can facilitate indexing, searching and retrieving of models, and thus their management and reuse. The adoption of IR techniques on source code dates back to the early 2000s, and within the MDE community there has been some recent effort in this direction (e.g. by Bislimovska et al. [7]). Further IR-based techniques can be found in [2,4] involving repository management and model searching scenarios.

Natural Language Processing. Accurate Natural Language Processing (NLP) is needed to handle realistic models with noisy text content, compound words, and synonymy/polysemy. In our experience, it is very problematic to blindly use NLP tools on models, e.g. just WordNet synonym checking without proper part-of-speech tagging and word sense disambiguation. More research is needed to find the right chain of NLP tools applicable for models (in contrast to source code and documentation), and reporting accuracies and disagreements between tools (along the lines of the recent report in [20] for repository mining). Note that NLP offers further advanced tools, such as language modelling, which are still to be investigated for MDE.

Data Mining. Following the perspective of approaching MDE artefacts as data, we need scalable techniques to extract relevant units of information from models (*features* in data mining jargon), and to discover patterns including domain clusters, outliers/noise and clones. Several example applications can be found in [1,2,4]. To to analyse, explore and eventually make sense of the large datasets in MDE (e.g. the Lindholmen dataset), we can investigate what can be borrowed from comparable approaches in data mining for structured (graph) data.

Machine Learning. The increasing availability of large amounts of MDE data can be exploited, via machine learning, to automatically infer certain qualities and functions. There has been a thrust of research in this direction for source code (e.g. for fault prediction [10]), and it would be noteworthy to investigate the emerging needs of the MDE communities and feasibility of such learning techniques for MDE. The approach in [3] for learning model transformations by examples is one of the few pieces of such work in MDE.

Visualization. We propose visualization and visual analytics techniques to inspect a whole dataset of artefacts (e.g. cluster visualizations in [4], in contrast with visualizing a single big model in [18]) using various features such as metrics and cross-artefact relationships. The goals could range from exploring a repository to analysing an MDE ecosystem holistically and even studying the (co-)evolution of MDE artefacts.

Distributed/Parallel Computing. With the growing amount of data to be processed, employing distributed and parallel algorithms in MDE is very relevant. There are conceptually related approaches in MDE worthwhile investigating, e.g. distributed model transformations for very large models [6,9] or model-driven data analytics [13]. Yet we wish to draw attention here to performing computationally heavy data mining or machine learning tasks for large MDE datasets in an efficient way.

We propose this non-exhaustive list as a preliminary exploitation guideline to help tackling scalability in MDE. Although the aforementioned domains themselves are quite mature on their own, it should be investigated to what extent results and approaches can be transferred into the MDE technical space.

5 Conclusion

We observe a rapid increase in the size of the MDE universe, both in open source and industry, which leads to scalability issues yet to be addressed by the community. To overcome this new and relatively overlooked challenge, we propose a holistic research perspective with several components, ranging from information retrieval to machine learning. We believe that approaches towards this direction already matter, but will increasingly be more important for the successful widespread use of MDE.

Acknowledgments. This work is supported by the 4TU.NIRICT Research Community Funding on Model Management and Analytics in the Netherlands.

References

1. Babur, Ö.: Statistical analysis of large sets of models. In: 31th IEEE/ACM International Conference on Automated Software Engineering (ASE 2016), Singapore, 3–7 September 2016 (2016)
2. Babur, Ö., Cleophas, L., van den Brand, M.: Hierarchical clustering of metamodels for comparative analysis and visualization. In: 2016 Proceedings of the 12th European Conference on Modelling Foundations and Applications, pp. 2–18 (2016)
3. Baki, I., Sahraoui, H.A.: Multi-step learning and adaptive search for learning complex model transformations from examples. ACM Trans. Softw. Eng. Methodol. **25**(3), 20:1–20:37 (2016). https://doi.org/10.1145/2904904
4. Basciani, F., Di Rocco, J., Di Ruscio, D., Iovino, L., Pierantonio, A.: Automated clustering of metamodel repositories. In: Nurcan, S., Soffer, P., Bajec, M., Eder, J. (eds.) CAiSE 2016. LNCS, vol. 9694, pp. 342–358. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39696-5_21
5. Basciani, F., Di Rocco, J., Di Ruscio, D., Di Salle, A., Iovino, L., Pierantonio, A.: MDEForge: an extensible web-based modeling platform. In: Proceedings of the 2nd International Workshop on Model-Driven Engineering on and for the Cloud Co-located with the 17th International Conference on Model Driven Engineering Languages and Systems, CloudMDE@MoDELS 2014, Valencia, Spain, 30 September 2014, pp. 66–75 (2014). http://ceur-ws.org/Vol-1242/paper10.pdf
6. Benelallam, A., Gómez, A., Tisi, M., Cabot, J.: Distributed model-to-model transformation with ATL on MapReduce. In: Proceedings of the 2015 ACM SIGPLAN International Conference on Software Language Engineering, pp. 37–48. ACM (2015)
7. Bislimovska, B., Bozzon, A., Brambilla, M., Fraternali, P.: Textual and content-based search in repositories of web application models. TWEB **8**(2), 11:1–11:47 (2014). https://doi.org/10.1145/2579991
8. Brambilla, M., Cabot, J., Wimmer, M.: Model-Driven Software Engineering in Practice, 1st edn. Morgan & Claypool Publishers, San Rafael (2012)
9. Burgueño, L., Wimmer, M., Vallecillo, A.: Towards distributed model transformations with LinTra (2016)
10. Catal, C., Diri, B.: A systematic review of software fault prediction studies. Expert Syst. Appl. **36**(4), 7346–7354 (2009)
11. Deissenboeck, F., Hummel, B., Juergens, E., Pfaehler, M., Schaetz, B.: Model clone detection in practice. In: Proceedings of the 4th International Workshop on Software Clones, pp. 57–64. ACM (2010)
12. Di Rocco, J., Di Ruscio, D., Iovino, L., Pierantonio, A.: Mining metrics for understanding metamodel characteristics. In: Proceedings of the 6th International Workshop on Modeling in Software Engineering, MiSE 2014, pp. 55–60. ACM, New York (2014). http://doi.acm.org/10.1145/2593770.2593774
13. Hartmann, T., Moawad, A., Fouquet, F., Nain, G., Klein, J., Traon, Y.L., Jezequel, J.M.: Model-driven analytics: connecting data, domain knowledge, and learning. arXiv preprint arXiv:1704.01320 (2017)
14. Hebig, R., Ho-Quang, T., Chaudron, M.R.V., Robles, G., Fernández, M.A.: The quest for open source projects that use UML: mining GitHub. In: Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems, Saint-Malo, France, 2–7 October 2016, pp. 173–183 (2016). http://dl.acm.org/citation.cfm?id=2976778

15. Hotho, A., Nürnberger, A., Paass, G.: A brief survey of text mining. LDV Forum **20**(1), 19–62 (2005). http://www.jlcl.org/2005_Heft1/19-62_HothoNuernbergerPaass.pdf
16. Kagdi, H., Collard, M.L., Maletic, J.I.: A survey and taxonomy of approaches for mining software repositories in the context of software evolution. J. Softw. Maint. Evol. **19**(2), 77–131 (2007). https://doi.org/10.1002/smr.344
17. Kolovos, D.S., Matragkas, N.D., Korkontzelos, I., Ananiadou, S., Paige, R.F.: Assessing the use of eclipse MDE technologies in open-source software projects. In: OSS4MDE@ MoDELS, pp. 20–29 (2015)
18. Kolovos, D.S., Rose, L.M., Matragkas, N., Paige, R.F., Guerra, E., Cuadrado, J.S., De Lara, J., Ráth, I., Varró, D., Tisi, M., Cabot, J.: A research roadmap towards achieving scalability in model driven engineering. In: Proceedings of the Workshop on Scalability in Model Driven Engineering, BigMDE 2013, pp. 2:1–2:10. ACM, New York (2013). http://doi.acm.org/10.1145/2487766.2487768
19. Mengerink, J.G., Serebrenik, A., Schiffelers, R.R., van den Brand, M.G.: Automated analyses of model-driven artifacts. IWSM Mensura (2017, to appear)
20. Omran, F.N.A.A., Treude, C.: Choosing an NLP library for analyzing software documentation: a systematic literature review and a series of experiments. In: Proceedings of the 14th International Conference on Mining Software Repositories, MSR 2017, Buenos Aires, Argentina, 20–28 May 2017, pp. 187–197 (2017). https://doi.org/10.1109/MSR.2017.42
21. Roy, C.K., Cordy, J.R., Koschke, R.: Comparison and evaluation of code clone detection techniques and tools: a qualitative approach. Sci. Comput. Program. **74**(7), 470–495 (2009). http://www.sciencedirect.com/science/article/pii/S0167642309000367
22. Störrle, H.: Towards clone detection in UML domain models. Softw. Syst. Model. **12**(2), 307–329 (2013)
23. Störrle, H., Hebig, R., Knapp, A.: An index for software engineering models. In: Joint Proceedings of MODELS 2014 Poster Session and the ACM Student Research Competition (SRC) Co-located with the 17th International Conference on Model Driven Engineering Languages and Systems (MODELS 2014), Valencia, Spain, 28 September–3 October 2014, pp. 36–40 (2014). http://ceur-ws.org/Vol-1258/poster8.pdf
24. Whittle, J., Hutchinson, J., Rouncefield, M.: The state of practice in model-driven engineering. IEEE Softw. **31**(3), 79–85 (2014)