Research directions for agile model-driven engineering

Kevin Lano[1], H. Alfraihi[2] and H. Haughton[3]

[1]King's College London, Strand, London, UK
[2]Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia
[3]Holistic Risk Solutions Ltd, London, UK

Abstract

Model-driven Engineering (MDE) advocates software development based on the use of software models and model transformations. Despite extensive research, and many cases of MDE application across a range of application domains, MDE still remains a minority practice among software developers. Agile MDE aims to combine MDE techniques and processes with those of agile methods such as Scrum, and has achieved significant adoption within certain industries, such as automotive systems and telecoms. Based on recent empirical studies and on a survey of 349 papers over the last 25 years of Agile MDE research, we highlight key issues with the current use of MDE, and identify where Agile MDE approaches can contribute to advance MDE practice.

Keywords

Model-driven Engineering, Agile Development, Systematic Literature Review

1. Introduction

The ideas behind model-driven engineering (MDE) originated in the 1990's, from work on the integration of object-oriented modelling and formal methods, and were principally motivated by the aim of increasing the rigour of object-oriented development [1, 2, 3]. The concepts of MDE were explicitly defined in the Model-driven architecture (MDA) from the OMG, which emphasised modelling at different levels of abstraction, together with model transformations (MT) to support automated mapping between models, and from models to code[1]. The MDE field has substantially expanded and developed over the last 25 years, however concerns continue to be raised regarding the high resources and skills required to utilise MDE for industrial software development [4, 5, 6]. Section 2 summarises current MDE practice and issues with the use of MDE.

Agile methods have been combined with MDE in 'Agile MDE' approaches, which aim to gain benefits for agile and/or MDE development approaches by adopting techniques from both methods. Here we provide an updated survey of the agile MDE field, which has developed

AMDE 2024: Agile Model-driven Engineering Workshop, Part of the Software Technologies: Applications and Foundations (STAF) federated conferences, Eds. S. Tehrani, H. Alfraihi, S. Rahimi and J. Troya, 8–12 July 2024, Enschede, Netherlands.

kevin.lano@kcl.ac.uk (K. Lano); haalafraihi@pnu.edu.sa (H. Alfraihi); h.haughton@gmail.com (H. Haughton)
0000-0002-9706-1410 (K. Lano)

[1]www.omg.org/mda/specs.htm

significantly since the survey of [7]. In Section 3 we employ the systematic literature review (SLR) procedure of [8], and in Section 4 use the survey results to propose specific research directions whereby Agile MDE concepts can contribute to improve MDE practice and adoption.

2. Current state of MDE practice

In this section we analyse the current situation regarding MDE adoption and practice, and identify issues which have been highlighted by empirical research and raised by MDE practitioners in forums at recent conferences.

We consider the following MDE survey papers from the last 7 years: [4, 5, 6]. Table 1 summarises the findings and recommendations of these papers.

The findings of these surveys about MDE benefits and issues are generally consistent: it is agreed that MDE can have significant benefits in terms of software quality and development productivity and rigour. However, gaining these benefits requires various obstacles to be overcome, particularly due to the effort and specialised skills needed to effectively use MDE languages and tools. There is a lack of appropriate or usable tools, and poor integration between MDE tools and between MDE processes and mainstream software engineering practices.

The above findings are also consistent with the results of more specialised surveys, such as [9, 10, 11] and [12, 13]. In [11], a large-scale interview study was conducted to identify the benefits/disadvantages of model transformation languages (MTLs) compared to general-purpose languages (GPLs) for writing model transformations. While positive benefits of using MTLs in terms of productivity of transformation development, and for conciseness and clarity of transformations were described, these were often counterbalanced by the need for specialised skills and tools, and weaknesses of MTLs in areas such as transformation reuse, learnability and poor tool support. GPLs were perceived to generally provide much stronger tooling, interoperability and ecosystem support compared to MTLs.

From the above surveys it can be seen that the potential benefits of MDE are widely recognised, but have been limited by various factors – of which *Poor MDE process integration*; *Lack of appropriate tools (especially for requirements analysis and maintenance)*; *Poor usability and high skill requirements of MDE languages and tools* have been repeatedly mentioned by multiple empirical studies.

Subsequent to the survey and roadmap of [6], the landscape of software engineering and the specific MDE field have changed in some significant ways:

- The SysML 2.0 standard has been published[2]. Driven by industry concerns about the deficiencies of UML and SysML, this is a substantial revision of SysML which emphasises traceability and textual modelling.
- The *digital twins* field has emerged as a means of applying modelling concepts for the simulation of real-world situations and artefacts in health, urban environments and many other application areas [14, 15, 16].
- Cloud-based and web-based applications have become a principal means for supporting software development and deployment. These applications include low-code development platforms [17].

[2] github.com/Systems-Modeling/SysML-v2-Release/

Table 1
Summary of MDE surveys

Survey	*MDE benefits*	*MDE issues*	*Suggested solutions*
Abrahao et al [5] (2017)	Abstraction; Automatic code generation.	Poor UX.	Move to user-driven MDE.
		Lack of MDE integration (within MDE & between MDE and other development practices).	Improved interoperability & integration.
		High skills needed to use MDE processes/tools.	Improve/evaluate UX to reduce cognitive load of tools.
		Complex languages & tools.	First class support for customisation & domain-specific modelling.
		Access to MDE skills.	
Bucchiarone et al, [6] (2020)	Abstraction; Automation.	Legacy models; Uncertainty in design; Dealing with software change.	Improved MDE agility and maintenance support.
		Marginal adoption of MTL by industry.	Improved support for bidirectionality.
		Challenges of IoT, smart systems.	Domain-specific MDE.
		Complexity of graphical modelling tools; Linking different views.	Increase adoption of textual modelling.
		Scalability; Accidental complexity of MDE.	Use of AI to support more powerful MDE tools.
		Limited end-user tailoring capabilities of MDE.	Deeper use of example-based modelling.
		Education issues.	Model repositories; Improved MDE teaching.
Alfraihi & Lano [4] (2023)	Improved software quality; Consistency of models & code; Improved communications across teams; Improved reusability.	Manual processes used for requirements analysis & model creation; Lack of tool support for requirements specification, testing, operations, deployment or maintenance.	More flexible & customisable tools/ notations.
		Lack of skills; Steep learning curve; Lack of integration.	Emphasis on learnability & integration; User evaluation for MDE tools.

- Perhaps most significant, the field of large language models (LLMs) [18] within machine learning (ML) has emerged as a radical advance on previous ML capabilities, and there is a rapid and ongoing growth in the application of LLMs to SE and to MDE [19, 20].

3. Research in agile MDE

Agile MDE denotes the combination of agile methods [21] and MDE. Agile MDE approaches were proposed soon after the inception of MDE, and have been used in a wide range of industrial settings. To examine the evolution of Agile MDE, we carried out a systematic literature review (SLR) of publications in the Agile MDE field, using the standard SLR procedure of [8]. The research questions we aimed to answer from the survey were:

RQ1: How has the Agile MDE field evolved and developed over time? In particular:

1. Have there been changes in the application areas considered?
2. Have there been changes in the agile or MDE techniques used?
3. Have there been changes in the industrial versus academic orientation of papers?

RQ2: Have any 'best practices' for Agile MDE emerged?

We performed a search on the three leading academic online libraries: Scopus, IEEE Xplore and ACM Digital Library, using the search criteria:

1. Peer-reviewed conference or journal papers.
2. Must include both 'agile' and 'model-driven' in the abstract.
3. In the date range 1999–2024.

The search generated 1407 candidate papers divided as follows: Scopus – 350, IEEE Xplore – 80, ACM Digital Library – 977. All survey data is at: zenodo.org/records/11611376.

After removal of duplicate papers, and inspection of abstracts and contents to exclude irrelevant publications, the total number of articles was reduced to 349 unique relevant papers.

Table 2 summarises the highlights of the data analysis of these papers in terms of *RQ*1 (1) and (2), and Figure 1 gives a chart of the number of publications in the field per year[3]. This shows a generally increasing trend in the number of publications.

With regard to *RQ*1 (3), there is a relatively high proportion of 'industrial' papers in the survey results, that is, papers with at least one industry-based co-author and/or including a real-world case. Overall, 38% of the papers (131) can be classified as industrial in this sense, and this percentage has increased from 30% in the period 1999–2010 to 41% in 2011–2020 and 40% in 2021–2024.

With regard to *RQ*2, there has been extensive research on techniques which could be used to align MDE and agile processes, such as rapid and evolutionary prototyping [22, 23], model and code co-evolution [24], 'low code' development [25, 26, 27], increased customer involvement in development [28, 29, 30, 31], and the automated generation of models from code or from natural language [32, 33], with a particular focus on the generation of models from user stories [34, 35, 36, 37]. These techniques can all potentially benefit MDE practice by accelerating development steps and reducing the manual effort required to create and manage models.

[3]Only complete years with at least one publication are included.

Table 2
RQ1: Agile MDE publication characteristics

Period	*Papers*	*Academic/ Industrial*	*Main application areas*	*Technical Areas*
1999–2010	84	55/24	Mechatronics/CPS 17 Web/SOA 15 Software Eng. 10 Business/finance 5	Code generation 10 Lightweight MDE 10 Architecture 9 Testing 7
2011–2020	211	124/86	Software Eng. 43 CPS/Embedded 34 Web/SOA 32 Business/finance 30	Requirements Eng. 21 DSLs 18 Testing/TDD 15 Code generation 13
2021–2024	54	31/21	Software Eng. 15 Business/finance 12 CPS/Embedded 10	Requirements Eng. 6 Lightweight MDE 5 DevOps/CI 4

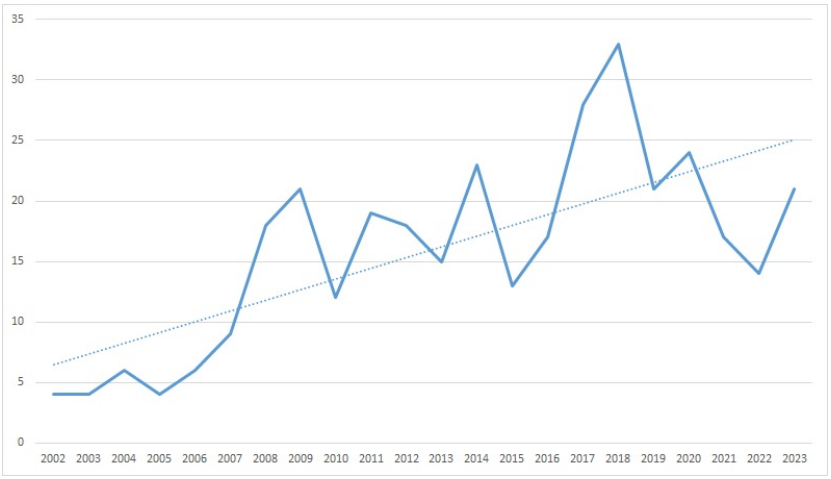

Figure 1: RQ1: Agile MDE papers published per year

4. Research directions for MDE improvement

The analysis of Section 2 identified problems with MDE adoption in three categories:

- *MDE process issues*: e.g., lack of integration with SE practices.
- *MDE language issues*: language complexity and difficulty in learning the languages.
- *MDE tool issues*: complex tool interfaces, poor usability, lack of customisability of tools.

Taking into account the above analysis, and the work in the field of agile MDE described in Section 3, we can point out specific research directions which have the potential to improve the uptake and effectiveness of MDE:

- Simplified and lightweight MDE languages, particularly the use of textual modelling languages instead of graphical, and transformations specified in terms of concrete syntax instead of metamodel features.

- Usability improvement, by means of user co-design of MDE languages and tools, and formal usability analysis of produced tools/languages.
- Simplified and lightweight tools, using a 'low-code' style of interaction and accessible via web or mobile UIs.
- Process and tool use support for users, including requirements formalisation and applications of machine learning for modelling guidance and transformation construction from examples.
- Improved integration with SE, in areas such as reverse-engineering/legacy code management (model-driven reverse engineering, MDRE), software architecture, CI/CD/DevOps and open-source/collaborative development platforms.

These areas are summarised in Figure 2.

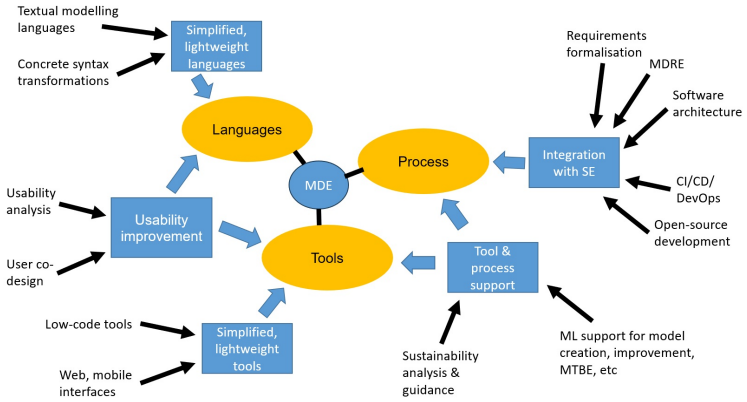

Figure 2: MDE roadmap directions

4.1. Simplified and lightweight MDE languages

The complexity of graphical modelling languages such as UML have often been cited as an obstacle to software modelling use. Large graphical models can require significant effort to comprehend or manage, and text-based models are often more convenient for operations such as model management, model merging, and for model comparison using tools such as *diff*, etc [13]. There is some empirical evidence that the effort and time needed for understanding textual models is lower than for graphical [38, 39]. Thus further research on text-based modelling approaches and tools is a potentially useful research direction.

Model creation is a critical activity in MDE, which currently requires high effort and expertise. Model creation effort could be reduced by (i) using requirements-to-model synthesis approaches (requirements formalisation) based on natural language processing or machine learning [40]; (ii) reverse-engineering models from existing code using MDRE [41, 42]; (iii) using 'sketch-to-model' techniques to produce a formal model from informal sketches, analogous to 'sketch-to-code' tools[4].

[4]www.microsoft.com/en-us/ai/ai-lab-sketch2code

With regard to DSL tooling, the use of metamodelling and model transformations relying on metamodels may complicate and increase the effort of DSL support. Instead, a more lightweight text- and grammar-based approach using tools such as ANTLR has been recommended [43, 44] and should be further investigated. Transformations could be specified based on the *concrete syntax* of source and target languages, using a text-to-text MTL such as $\mathcal{CSTL}$ [45], thus avoiding the need for transformation developers to know the metamodels of the languages.

4.2. User-centered languages and tools

MDE languages and tools should be designed with respect to the needs and capabilities of potential users, and evaluated with respect to such user needs. Ideally, experts from a domain, representative of end-users in that domain, should play a leading/directing role in the development of MDE language and tool support for that domain. Thus, agile methods concepts of an 'on-site customer representative' should be adopted [46]. An example of this process was the creation of a DSL, MathOCL, for the specification of mathematical models in finance [47]. This work was initiated and guided by a domain expert, working together with MDE experts.

MDE tool and language creators should consider and evaluate the cognitive loading of their products, and avoid factors which increase this load, such as novel or non-standard concepts and syntax. Aligning tools and languages to existing user skills and knowledge can help to overcome obstacles to use.

4.3. Machine learning and MDE

An area with rapidly increasing research activity is the application of AI and machine learning (ML) to software engineering [20]. Within MDE, ML has been used to (potentially) improve the user experience by simplifying and automating MDE processes such as the construction of models from informal requirements [48, 40, 49, 50], and the construction of model transformations from examples [51, 45]. ML has also been used for advisory systems to assist practitioners to analyse and improve models, and to support model-based DevOps processes [52]. In principle an LLM with sufficiently wide-ranging knowledge could be used as an 'operating system' or platform able to provide specialised knowledge-based services, including software modelling services, after suitable fine-tuning [19]. However there can be reliability and accuracy issues with non-symbolic machine learning techniques such as LLMs, which means that their outputs need to be considered as the views of a fallible and sometimes inconsistent expert, rather than being used to directly automate processes [53, 54, 55]. Current LLMs also have inadequate modelling knowledge to be directly used to automatically support MDE activities [48, 40, 49]. Thus a human modeller needs to inspect, correct and validate the models produced by an LLM.

We consider that an important research direction to pursue is the creation of new LLMs possessing sufficient code, natural language and modelling knowledge in order to more effectively support MDE activities such as model creation from requirements, model review, and model quality improvement. In turn, this work will require the creation and curation of substantial model datasets to support training of the LLMs.

Further investigation should be carried out into the use of *symbolic* ML for learning precise software engineering tasks, such as model transformation (MT) and code generation and

abstraction [56, 45, 41].

4.4. Simplified and lightweight tools

Low-code software development approaches are techniques whereby end-users can construct applications with a minimum of traditional coding. They typically utilise a pre-existing set of services (such as data persistence services) provided by a cloud-based platform. There is some overlap between low-code and MDE approaches [17], because MDE approaches usually involve the replacement of explicit coding by more abstract/platform-independent specification and design. But many low-code approaches do not involve construction of models, instead drag-and-drop facilities as in interface builders such as Xcode are used. In addition, while a goal of low-code is to reduce the development effort and skills needed to produce an application, MDE approaches instead move effort from code construction to model construction, and require advanced skills. Our view is that the low-code approach would be useful for MDE tools that aim to support high user configurability, or to enable users with lower MDE technical skills to utilise MDE techniques. In this guise they would actually be 'low modelling and low code' tools, enabling the construction of model-based solutions using pre-existing components such as OCL libraries for file management and networking [57]. ML support could be used to automate the selection of model elements and design and architecture choices from user input provided in natural language. Speech input/output could be used for improved interactivity.

4.5. Integration with software engineering practices

Surveys have emphasised the need for MDE techniques to be seamlessly integrated with existing development practices such as continuous integration/deployment and DevOps [5, 4]. The selective use of MDE techniques to enhance agile development processes has been a major theme in the Agile MDE field (Section 3), with modelling being used to facilitate communication between heterogeneous teams [13], and to accelerate the production of prototypes [22, 23]. A major challenge in MDE and mainstream SE integration is how to combine automated code generation with manually-produced code, including legacy code. Prospective solutions include model and code co-evolution [24] and automated extraction of models from code [42].

The use of MDE with open-source projects involving collaborative development is also an area where more work is needed [33]. The study of [58] showed that MDE use may have positive benefits in the early life of such projects, but this effect diminishes over time, which may be due to the general lack of effort to synchronise models and code, so that the models become progressively more out-of-date over time as project contributors make code changes.

5. Conclusions

This paper has distilled the analysis of recent surveys of the MDE field, and highlighted significant work from the field of Agile MDE in order to identify key topics for research to improve MDE usability and the acceptance of MDE by general software practitioners.

In summary, we identified research focus areas of simpler and more lightweight MDE languages, tools and processes, with increased user involvement in MDE language and tool defini-

tion and evaluation, and increased automation support for MDE processes, using ML or other techniques. Improved integration with SE processes is also needed. We consider that these research directions have potential to expand the use of MDE and to increase the benefits from such use, and hence ultimately to improve the practice of software engineering.

References

[1] S. Cook, J. Daniels, Designing Object Systems: Object-oriented modelling with Syntropy, Prentice Hall, 1994.

[2] G. Rose, Object-Z, in: Object orientation in Z, 1992, pp. 59–77.

[3] J. P. Bowen, P. Breuer, K. Lano, Formal specifications in software maintenance: from code to Z++ and back again, Information and Software Technology 35 (1993) 679–690.

[4] H. Alfraihi, K. Lano, Trends and insights into the use of model-driven engineering: a survey, in: SAM/MODELS, 2023.

[5] S. Abrahão, F. Bourdeleau, B. Cheng, S. Kokaly, R. Paige, H. Stöerrle, J. Whittle, User experience for model-driven engineering: Challenges and future directions, in: 2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS), IEEE, 2017, pp. 229–236.

[6] A. Bucchiarone, J. Cabot, R. Paige, A. Pierantonio, Grand challenges in MDE: an analysis of the state of the research, SoSyM 19 (2020) 5–13.

[7] H. A. A. Alfraihi, K. C. Lano, The integration of agile development and model driven development: A systematic literature review, The 5th International Conference on Model-Driven Engineering and Software Development (2017).

[8] B. Kitchenham, S. Charters, Guidelines for performing systematic literature reviews in software engineering, 2007. Tech. Report EBSE 2007-001, Keele University.

[9] D. Akdur, V. Garousi, O. Demirörs, A survey on modeling and model-driven engineering practices in the embedded software industry, Journal of Systems Architecture 91 (2018) 62–82.

[10] L. Burgueño, J. Cabot, S. Gérard, The future of model transformation languages: An open community discussion, Journal of Object Technology 18 (2019).

[11] S. Hoppner, Y. Haas, M. Tichy, K. Juhnke, Advantages and disadvantages of (dedicated) model transformation languages, Empirical Software Engineering 27 (2022).

[12] G. Liebel, M. Tichy, E. Knauss, Use, potential and showstoppers of models in automotive requirements engineering, Software & Systems Modeling 18 (2019) 2587–2607.

[13] G. Liebel, E. Knauss, Aspects of modelling requirements in very-large agile systems engineering, Journal of Systems and Software 199 (2023).

[14] F. Bordeleau, B. Combemale, R. Eramo, M. van den Brand, M. Wimmer, Towards model-driven digital twin engineering: current opportunities and future challenges, in: Systems Modelling and Management, Springer, 2020, pp. 43–54.

[15] R. Laubenbacher, J. Sluka, J. Glazier, Using digital twins in viral infection, Science 371 (2021) 1105–1106.

[16] L. Cleophas, et al., Model-driven engineering of digital twins, 2022. Report from Dagstuhl seminar 22362.

[17] D. D. Ruscio, D. Kolovos, J. de Lara, A. Pierantonio, M. Tisi, M. Wimmer, Low-code development and MDE: two sides of the same coin?, Sosym 21 (2022) 437–446.

[18] W. Zhao, et al., A survey of large language models, arXiv 2303.18223v10 (2023).

[19] B. Combemale, J. Gray, B. Rumpe, Large language models as an 'operating' system for software and systems modeling, SoSym (2023) 1391–1392.

[20] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, H. Wang, LLMs for software engineering: a systematic literature review, arXiv 2308.10620 (2023).
[21] K. Beck, et al., Manifesto for agile software development, 2001. Agilemanifesto.org.
[22] F. P. Basso, R. M. Pillat, F. Roos-Frantz, R. Z. Frantz, Combining MDE and Scrum on the rapid prototyping of web information systems, International Journal of Web Engineering and Technology 10 (2015) 214–244.
[23] M. Sedlmeier, M. Gogolla, A toolchain transforming descriptive domain-specific models into executable browser-based applications, in: CEUR Workshop Proceedings, volume 2716, 2020, pp. 142–149.
[24] C. Bernaschina, E. Falzone, P. Fraternali, S. L. H. Gonzalez, The virtual developer: Integrating code generation and manual development with conflict resolution, ACM Transactions on Software Engineering and Methodology 28 (2019).
[25] S. Bosselmann, M. Frohme, D. Kopetzki, M. Lybecait, S. Naujokat, J. Neubauer, D. Wirkner, P. Zweihoff, B. Steffen, DIME: A programming-less modeling environment for web applications, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 9953 LNCS (2016) 809–832.
[26] C. I. Gog, Agile development of PHP websites: A model-aware approach, Complex Systems Informatics and Modeling Quarterly 2020 (2020) 19–31.
[27] J. M. Rivero, J. Grigera, D. Distante, F. Montero, G. Rossi, DataMock: An agile approach for building data models from user interface mockups, Software and Systems Modeling 18 (2019) 663–690.
[28] D. Aveiro, V. Freitas, E. Cunha, F. Quintal, Y. Almeida, Traditional vs. low-code development: Comparing needed effort and system complexity in the NexusBRaNT experiment, in: Proceedings 2023 IEEE 25th Conference on Business Informatics, CBI 2023, 2023.
[29] M. Bragilovski, F. Dalpiaz, A. Sturm, From user stories to domain models: Recommending relationships between entities, in: CEUR Workshop Proceedings, volume 3378, 2023.
[30] M. Wahler, E. Conte, M. Frick, D. Mosquera, M. Ruiz, Rapid software prototyping from business artifacts, in: CEUR Workshop Proceedings, volume 3134, 2022, pp. 9–14.
[31] L. Westermann, J. Mey, R. Schone, U. Assmann, A performant low-code system for the timely implementation of road safety regulations, in: Proceedings 2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, MODELS-C 2023, 2023, pp. 906 – 910.
[32] A. Boronat, Code-first model-driven engineering: On the agile adoption of MDE tooling, in: Proceedings 2019 34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019, 2019, pp. 874–886.
[33] F. Gilson, D. Weyns, When natural language processing jumps into collaborative software engineering, in: Proceedings 2019 IEEE International Conference on Software Architecture - Companion, ICSA-C 2019, 2019, pp. 238–241.
[34] K. Baina, M. El Hamlaoui, H. Kabbaj, Business process modelling augmented: Model driven transformation of user stories to processes, in: ACM International Conference Proceeding Series, 2020, pp. 215–220.
[35] F. Gilson, M. Galster, F. Georis, Generating use case scenarios from user stories, in: Proceedings 2020 IEEE/ACM International Conference on Software and System Processes, ICSSP 2020, 2020, pp. 31–40.
[36] T. Gunes, F. B. Aydemir, Automated goal model extraction from user stories using NLP, in: Proceedings of the IEEE International Conference on Requirements Engineering, volume 2020-August, 2020, pp. 382–387.
[37] S. Nasiri, Y. Rhazali, M. Lahmer, N. Chenfour, Towards a generation of class diagram from user stories in agile methods, Procedia Computer Science 170 (2020) 831–837.
[38] A. Ottensooser, et al., Making sense of business process descriptions: an experimental comparison

of graphical and textual notations, SoSym 85 (2012) 596–606.
[39] Z. Sharafi, et al., An empirical study on the efficiency of graphical vs textual representations in requirements comprehension, in: ICPC 2013, IEEE, 2013.
[40] J. Camara, J. Troya, L. Burgueno, A. Vallecillo, On the assessment of generative AI in modeling tasks, SoSyM 22 (2023).
[41] K. Lano, H. Haughton, Z. Yuan, H. Alfraihi, Program abstraction and re-engineering: an Agile MDE approach, in: SAM/MODELS 2023, 2023.
[42] K. Lano, H. Siala, Using MDE to automate software language translation, Automated Software Engineering 31 (2024).
[43] J. Cabot, Learning ANTLR – a software modeling perspective, 2018. Https://modeling-languages.com/learning-antlr-software-modeling/, accessed Dec. 2023.
[44] K. Lano, Q. Xue, Lightweight software language processing using ANTLR and CGTL, in: Proceedings of the 11th International Conference on Model-Driven Engineering and Software Development (MODELSWARD), 2023.
[45] K. Lano, Q. Xue, Code generation by example using symbolic machine learning, Springer Nature Computer Science (2023).
[46] R. Hoda, N. Salleh, J. Grundy, The rise and evolution of agile software development, IEEE Software (2018).
[47] H. Haughton, S. Y. Tehrani, K. Lano, MathOCL: a domain-specific language for financial modelling, in: Agile MDE Workshop, STAF 2023, 2023.
[48] S. Abukhalaf, M. Hamdaqa, F. Khomh, On Codex prompt engineering for OCL generation: an empirical study, arXiv:2303.16244v1 (2023).
[49] H.-G. Fill, P. Fettke, J. Kopke, Conceptual modeling and large language models: Impressions from first experiments with ChatGPT, Enterprise Modelling and Information Systems Architectures 18 (2023).
[50] M. Umar, K. Lano, Advances in automated support for requirements engineering: a systematic literature review, Requirements Engineering (2024) 1–31.
[51] L. Burgueno, J. Cabot, S. Gerard, An LSTM-based neural network architecture for model transformations, in: MODELS '19, 2019, pp. 294–299.
[52] R. Eramo, et al., AIDOaRt: AI-augmented automation for DevOps, a model-based framework for continuous development in cyber-physical systems (2023).
[53] Y. Liu, C. Tantithamthavorn, Y. Liu, L. Li, On the reliability and explainability of automated code generation approaches, arXiv:2302.09587v1 (2023).
[54] A. Malyaya, et al., On ML-based program translation: perils and promises, arXiv:2302.10812v1 (2023).
[55] S. Ouyang, J. Zhang, M. Harman, M. Wang, LLM is like a box of chocolates: the non-determinism of ChatGPT in code generation, arXiv 2308.02828v1 (2023).
[56] K. Lano, S. Kolahdouz-Rahimi, S. Fang, Model Transformation Development using Automated Requirements Analysis, Metamodel Matching and Transformation By-Example, ACM TOSEM 31(2) (2021) 1–71.
[57] K. Lano, S. Kolahdouz-Rahimi, K. Jin, OCL libraries for software specification and representation, in: OCL 2022, MODELS 2022 Companion Proceedings, 2022.
[58] O. Baddreddin, K. Rahad, The impact of design and UML modeling on codebase quality and sustainability, in: Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering, CASCON '18, IBM Corp., USA, 2018, p. 236–244.