Towards AI-Enabled Model-Driven Architecture: Systematic Literature Review

Zina Zammel, Mouna Rekik, Lotfi Souifi and Ismael Bouassida Rodriguez
ReDCAD Laboratory, ENIS, University of Sfax, Tunisia

Keywords: Artificial Intelligence, Systematic Literature Review, Model Driven Architecture, CIM, PIM, PSM, Code Generation.

Abstract: The convergence of two separate areas of computer science, like Model Driven Architecture and Artificial Intelligence, can lead to collaboration in two main ways, such as AI-driven MDA and MDA for AI. In this paper, we present a Systematic Literature Review (SLR) on the application of AI within MDA. Additionally, we examine how AI facilitates transformations between the Computation Independent Model (CIM), Platform Independent Model (PIM), and Platform Specific Model (PSM), highlighting methods that bridge conceptual models with technical specifications. This review contributes to a deeper understanding of AI's role in enhancing the effectiveness of MDA frameworks by analyzing existing studies that are selected using SLR. Based on a systematic search of IEEE, Science Direct, Springer, ACM, and google scholarrelevant articles published between 2018 and 2024 were identified. The adoption of AI introduces numerous benefits to software engineering, including enhanced support for designers and automation in model transformations.

1 INTRODUCTION

Implementing a software solution to meet business needs is a complex process that involves several steps. The first step is to translate the stakeholders' needs into the requirements of the future system, usually in a specification written in natural language. The Model Driven Architecture (MDA) approach plays an important role in addressing software complexity, ensuring consistency between different levels of system design, and facilitating application maintenance and evolution of applications (Zouani and Lachgar, 2024). Business Process Modeling Notation (BPMN) and UML are universally accepted standards for designing models in software development process using MDA.

MDA advocated by the Object Management Group (OMG) to highlight the importance of abstract mod- eling (Zouani and Lachgar, 2024). MDA defines three primary types of models : CIM represents the system's requirements and business context without describing the structure or processing ; PIM spec- ifies the structure and functionality of the system namely, models abstracts the details of PSM that provides technical information on the implementa- tion of the system using a particular technology or platform. In MDA, the requirements specified in a CIM must be traceable to the constructs in the PIM and the PSMs that implement them. Further- more, MDA facilitates a model-driven software de- velopment process through Model-to-Model (M2M) transformations, CIM requirements are transformed into PIM, which focuses on software functionality rather than implementation details. A Model-to-Text (M2T) transformation converts PIM models into PSM or source code. Acceleo, Xtend, EGL, TextGen, and AdoScript are common M2T languages, while ATL, EGL, and QVT-Operational are commonly used for M2M transformations. Utilizing MDA signifi- cantly lowers software development costs compared to conventional Software Development Life Cycle approaches, while maintaining high quality since code is generated from the established models. Moreover, Domain-Specific Models (DSM) enhance MDA) by enabling the creation of abstract, domain-focused representations that capture system requirements and designs across various domains. DSM utilizes two modeling notations: graphical, as in Domain-Specific Modeling Languages (DSML), and textual, as in Domain-Specific Languages (DSL). Despite its benefits, DSM faces challenges such as domain-specific customization, maintaining consistency across ab-

Zammel, Z., Rekik, M., Souifi, L. and Rodriguez, I. B.
Towards AI-Enabled Model-Driven Architecture: Systematic Literature Review.
DOI: 10.5220/0013374000003890

In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 3*, pages 1380-1387
ISBN: 978-989-758-737-5; ISSN: 2184-433X

straction levels, and managing transformations into platform-specific implementations. Semi-formal languages like UML, BPMN, and DSLs are commonly used to define PIMs, offering structured notations but lacking the precision of formal languages. This can result in ambiguities, unclear semantics, and overlap- ping interpretations, particularly in complex scenar- ios. On the other hand, extracting PIM involves con- verting system requirements, which are often written in natural language, into formal or semi-formal rep- resentations. This task has traditionally been the responsibility of humans due to the inherent complexity and ambiguity of natural language, making it challenging for machines to process. However, recent advancement in AI algorithms have shown promise in addressing these challenges by automating aspects of requirement analysis, Natural Language Processing (NLP), and model generation. Several systematic literature reviews on Model Driven Architecture have been conducted, such as the work by (Uzun and Tekinerdogan, 2018), which examined various Model Driven Architecture Based Testing approaches. Their SLR revealed that although MDABT is a generic process, the available approaches differed in their specific goals, modeling abstractions, and results. To the best of our knowledge, there is currently no existing SLR that focuses on the application of artificial intelligence within the context of MDA.

In this review, we explore how AI techniques have been applied to enhance the MDA process, specifically in overcoming challenges such as ambiguity of the requirements, consistency of the model, and automated transformation. In addition, we identify and discuss the limitations and opportunities these approaches present. This synthesis aims to provide researchers with insight to select appropriate AI algorithms, address persistent challenges, and explore future directions to advance the field of MDA. The remainder of this paper is structured as follows. In Section 2, we describe the adopted methodology for conducting our SLR. We describe the search strategy, the inclusion and exclusion criteria, and the data extraction process. Section 3 presents the results of our SLR and provides an analysis of the identified studies. In Section 4, we discuss the limitations of some proposed studies. In section 5, we present threats to validity of our SLR. In Section 6 we present a summary of our paper and our future direction.

2 RESEARCH METHOD

The research methodology for this study follows the SLR approach, comprising five steps: (i) Defining research objectives and questions, (ii) Conducting a literature search with targeted queries, (iii) Selecting studies using inclusion and exclusion criteria, (iv) Extracting data from selected studies, and (v) Analyzing results to address the research questions.

2.1 Research Questions

Identifying specific and valid research questions is the first step in SLR. To achieve the goal of this work, we aim to answer the following research questions (RQs):

- **RQ1** While using MDA what kind of AI algorithm can enhance designer work?
- **RQ2** How are AI algorithms used for CIM generation?
- **RQ3** How AI algorithms are used for PIM generation?
- **RQ4** How AI algorithms are used for PSM generation?
- **RQ5** How AI algorithms are used for Code generation?

2.2 Search String

A database search strategy was employed to col- lect relevant published literature, using systematic searches with well-defined search strings. The research string comprised keywords organized into three groups.

Group1: "model driven Architecture".

Group 2: "CIM", "PIM", PSM", "code genera- tion"

Group 3: "Artificial intelligence".

Both sets of keywords were combined with a Boolean search (AND, OR) in the article search process. The search string is reported below :

("Model Driven Architecture") AND ("PSM" OR "CIM" OR "CODE GENERATION" OR "PIM ")
AND ("Artificial intelligence")

2.3 Selection Criteria

Exclusion Criteria: For our SLR we excluded the Review papers (survey, SLR . . .), book chap- ters, master's, and Ph.D. are excluded. In addi- tion, publications that were published before or on 31.12.2017, and articles that were written in any language other than English are excluded.

Inclusion Criteria: Only peer-reviewed studies published in journals or conference proceedings were included. Also, we include the papers that contributed to solving MDA/ MDE challenges with AI in the abstract.

2.4 Data Extraction

The search string was used to collect the studies that are present in multiple sources. Specifically, the sources considered were IEEE, ACM, Science Direct, Springer and google scholar. The execution of the defined research query has selected 1261 articles to obtain 52 relevant studies following the application of the selection criteria. So, the outcomes of the selection process. 52 articles that matched the inclusion criteria were included in the SLR. we also included an additional 4 studies recommended by the expert. The distribution of selected studies according to the scientific databases and the publication type is presented in Table 1.

3 RESULTS AND ANALYSIS

This section presents the comprehensive results obtained from the conducted SLR. The results are organized and presented according to predefined inclusion and exclusion criteria, ensuring the relevance and quality of the selected studies.

3.1 RQ1: While Using MDA What Kind of AI Algorithm Can Enhance Designer Work?

(López et al., 2022) introduced ModelSet, a la- beled dataset of 5,466 Ecore meta-models and 5,120 UML models designed to advance machine learning in MDE. After removing non-English and uncategorized models, the dataset included 5,290 Ecore (14% ”dummy”) and 4,479 UML models (13% ”dummy”). Features with near-zero variance were eliminated, leaving 9 features for Ecore and 39 for UML. To address class imbalance, upsampling was applied, and 10-fold cross-validation with three repetitions optimized hyperparameters for classifiers, including k-NN, Random Forest, Neural Networks, and C5.0. Model performance was evaluated using paired t-tests for statistical significance. This work establishes ModelSet as a critical resource for AI-enhanced MDA, enabling tasks like model classification, tagging, and quality filtering while providing insights into feature relevance and classifier performance.

(Iyenghar et al., 2022) proposed integrating conversational AI frameworks, such as RASA, with MDE tools to facilitate guided tutorials, natural language query resolution, and dialog-based modeling support. They emphasized using AI techniques like machine learning for model transformation, semantic reasoning for DSM, and NLP to simplify modeling tasks, ultimately reducing complexity and the learning curve for MDA tools.

(Muttillo et al., 2024) proposed A novel MDE frame- work integrates event logs, intelligent modeling as- sistants (IMAs), and LLM generated modeling oper- ations to automate tasks and provide recommenda- tions. LLMs generate synthetic data to train IMAs, though human-based operations are more accurate. Deep learning techniques like LSTM networks pre- dict future modeling actions, fostering proactive de- sign. The authors evaluated the proposed framework in terms of correctness, diversity, and hallucination, showing that LLMs, particularly GPT-4, can effec- tively emulate human modeling operations. Based on the analysis, GPT-4 outperformed other LLMs across multiple metrics. It achieved a confidence interval (CI) entirely below 1, with lower bounds at 0.874 and upper bounds at 0.95, and an interquartile range (IQR) of 0, indicating minimal hallucination effects up to the 95th percentile. Additionally, GPT-4 exhib- ited the lowest standard error (0.0192), standard de- viation (0.352), and variance (0.124) among the com- pared models. A One-Sample Test with a null hy- pothesis that GPT-4’s mean is greater than 1 yielded a p-value $<$ 0.001, confirming GPT-4’s superior per- formance in minimizing hallucinations relative to the other models.

(Bruneliere et al., 2022) proposed innovative combination of MDE, NLP , and DevOps practices. This integration is intended to facilitate a smoother transition from design to runtime, thereby improving the overall efficiency of engineering processes in CPS. The role NLP is enhances requirement management by automating writing, ensuring consistency, aiding in elicitation, improving communication, and managing the lifecycle of requirements.

(Moin et al., 2022) presented a model-driven software engineering methodology that integrates DSM to enhance the development of AI-enabled IoT systems. It addresses challenges in traditional MDA, such as round-tripping and genericity, by focusing on deploying compact ML models on resource-constrained devices TinyML. This approach allows for local data processing, improving performance, availability and privacy. AI algorithms optimized for low-power environments, like decision trees and lightweight neural networks, can automate decision-making and enhance predictive capabilities.

Table 1: Summary of selected research studies and publication type.

Scientific database	Type	Studies	Total
IEEE	Journal	Not found	
	Conference	(Kulkarni et al., 2023), (Rigou et al., 2020) ,(Bhadra et al., 2024),(Lano and Xue, 2023),(Siala, 2024),(Zen- naro et al., 2018), (Thota et al., 2024), (Binder et al., 2021), (Benaben et al., 2019),(Tinnes et al., 2021), (Dorodnykh et al., 2018), (Houghtaling et al., 2024), (Moin et al., 2022), (Babaalla et al., 2024a), (Park et al., 2023),	15
ACM	Journal	Not found	
	Conference	(Yang and Sahraoui, 2022), (Safdar et al., 2022), (Kouissi et al., 2019), (Chang et al., 2020), (Babaalla et al., 2024b), (Uyanık and Sayar, 2023), (Sajji et al., 2023)	7
Springer	Journal	(Binder et al., 2022),(Mythily et al., 2019), (Panahandeh et al., 2021) , (Ouali et al., 2020), (Biswas et al., 2022), (Li et al., 2018), (Pe´rez-Castillo et al., 2022)	7
	Conference	Not found	
Science Direct	Journal	(Batchkova and Ivanova, 2019), (Maass and Storey, 2021), (Alulema et al., 2023), (Servadei et al., 2019)	4
Google Scholar	Journal	(Zouani and Lachgar, 2024),(Brandon et al., 2024) , (Bruneliere et al., 2022), (Eramo et al., 2024), (Khalfi et al., 2024),(Lo´pez et al., 2022), (Moin et al., 2021) , (Ouchra et al., 2024), (Tabbiche et al., 2023)	9
	Conference	, (Naveed et al., 2024), (Koseler et al., 2019) , (Sarazin et al., 2021), (Naimi et al., 2024), (Meyma et al., 2022), (Lopes et al., 2024), (Iyenghar et al., 2022), (Liu et al., 2020), (Muttillo et al., 2024)	9

(Bran- don et al., 2024) implemented CINCO de Bio, a low-code platform that simplifies biomedical imaging workflows by integrating model-driven architecture and AI. It enables non-technical users to design and execute computational workflows, offering modular- ity, scalability, and semantic validation. (Park et al., 2023) used MDA to facilitates the automation and ab- straction of the design process for neuromorphic ar- chitectures, enabling efficient exploration of hetero- geneous configurations and multi-objective optimiza- tion. AI plays a crucial role in guiding the design space search, allowing for the identification of op- timal architectural candidates based on performance metrics and user-defined constraints. (Eramo et al., 2024) proposed a novel architecture MDE, AI/ML, and DevOps automates processes like requirements, modeling, coding, testing, and monitoring. It lever- ages NLP for requirements analysis and GNNs for modeling insights, while AI/ML supports code gener- ation and reuse recommendations, enhancing system engineering and continuous delivery.

(Kulkarni et al., 2023) proposed a use case demon-strated how Generative AI, specifically ChatGPT, en- hances the MDE process by enabling domain experts to create models using natural language. Using in- puts such as a root goal, a meta-model description, and context, ChatGPT generates actionable strategies, like improving academic reputation and research out-put, as model instances. This approach bridges human intentions with technical models, streamlining MDE through natural language interactions while ensuring traceability and accuracy.

RQ3: How AI algorithms are used for PIM generation?

(Siala, 2024) Developed a model-driven reverse engi- neering approach tool using LLMs to generate UML and OCL specifications from source code because legacy systems become increasingly complex and more difficult to maintain, there is a growing need for new and better ways to understand and maintain them. Also, Class diagrams, which visually repre- sent classes, attributes, methods, and their interrela- tionships, play a pivotal role in capturing the core ar- chitecture of a software system. To address this need, (Sajji et al., 2023) proposed an approach that employs Graph Neural Networks, to automatically generate class diagrams from source code in the context of MDA and reverse engineering. According to the pa- per, software development company faces challenges with a complex, undocumented codebase, leading to longer development cycles and increased energy con- sumption. To address this, a Graph Neural Network (GNN) is used to generate class diagrams from the source code, improving system understanding. Focusing on a school management system in Java, the source code is analyzed to identify classes, attributes, methods, and relationships like inheritance and dependencies. A graph representation is constructed, with nodes for classes and edges for relationships, while relevant features are extracted. Trained on labeled datasets of code graphs and class diagrams, the GNN accurately captures class relationships and generates diagrams that enhance code clarity, streamline workflows, and reduce energy consumption.

Manually creating UML and use case diagrams can be tedious and error-prone, especially when the specifications are long and/or complex. As a result, (Babaalla et al., 2024a) suggested a novel method for examining textual specifications and extracting the relevant elements needed to create UML class and use case diagrams, utilizing NLP tools and linguistic techniques. Knowledge extraction module used for generating the concepts of the two UML diagrams of classes and use cases, from the output of the NLP module process used to analyze the text requirements. The elements of the two resulting diagrams are created using drawing algorithms and/or saved in XML format. The results demonstrated a high F1 score for the extraction of classes, attributes, and methods was reported to be in the interval of [80%; 100%]. In the same way (Yang and Sahraoui, 2022) presented a novel automated approach for generating UML class diagrams from natural language specifications.To develop this approach, they created a dataset of UML class diagrams and their English specifications. The pipeline of this work included several steps: segmenting the input text into sentences, classifying these sentences, generating UML class diagram fragments, and composing these fragments into a complete UML class diagram using natural language patterns and machine learning. They used Bernoulli Naive Bayes classifier binary classification of English sentences. The evaluation metrics results for classes are 17% precision and 25% recall for exact matching, the strictest metric. The results for relationships are a connectivity similarity of 63% and a size difference of 67%. (Tinnes et al., 2021) proposed OCKHAM, an unsupervised approach that learns domain-specific edit operations from model histories in repositories using frequent subgraph mining to identify meaningful patterns in model differences. The approach was evaluated on synthetic EMF models and a large-scale railway case study, demonstrating its ability to extract and recommend relevant edit operations in real-world settings. Furthrmore, (Rigou et al., 2020) Analyzed machine learning approaches for draft a PIM model that describes the functional requirements of a system from a textual specification.

(Tabbiche et al., 2023) proposed an intelligent meta- model that integrates the Eclipse Modeling Frame- work (EMF) with supervised machine learning to en- hance the adaptability and efficiency of the modeling process, particularly for ubiquitous applications. The approach employs multi-layer Perceptron (MLP) neu- ral networks to classify and predict outcomes based on contextual data, such as COVID-19 symptoms. A systematic process for PIM generation involves defin- ing a context meta-model and creating PIMs that are platform-independent. Using Acceleo for M2T trans- formation automates code generation for specific plat- forms, ensuring the relevance and adaptability of the generated models. The use case focuses on improv- ing COVID-19 patient classification using symptoms as input to an MLP-based neural network with one or two hidden layers. A dataset from the COVID- 19-TRACERSET repository by Public Health France serves as the training and test data. Initial KNN weights are randomly set and adjusted during learn- ing, where outputs above a threshold (> 0.5) are la- beled as "infected" (1) or "healthy" (0). Experimen- tal results demonstrate the reliability of ANN models in automatic decision-making, showcasing their accuracy in categorizing COVID-19 cases based on symptoms, ultimately improving the decision-making process.

3.2 RQ5 How AI Algorithms Are Used for Code Generation?

(Lano and Xue, 2023) applied novel symbolic machine learning techniques for learning tree-to-tree mappings of software syntax trees, to automate the development of code generators from source–target example pairs. This approach is referred to as Code Generation by Example. The method was evaluated across various tasks, including translating UML/OCL to programming languages such as Java, Kotlin, and C, as well as translating DSLs to SwiftUI. The results demonstrate both high accuracy and efficiency. (Bhadra et al., 2024) introduced a novel model-based code generator based on MDA to tackle the complexities of embedded programming, highlighting the need for diverse coding styles due to varying programming languages and hardware architectures. The authors demonstrate that their approach reduces the effort for generating low-level driver code for core tensor math operators in neural networks by an average of 62 times in Source Lines of Code compared to manual coding. This efficiency enhances the reliability and scalability of embedded software solutions, showcasing the value of model-driven methodologies in automating code generation. While acknowledging the potential of LLMs for code generation, the authors emphasize the deterministic outcomes of their approach, which ensures reliability and optimized code for embedded systems. Overall, the findings contribute significantly to software engineering and embedded systems development.

4 LIMITATIONS

AI has shown significant potential to enhance software engineering processes, but its application in MDA is still emerging and faces several limitations. A major challenge is the lack of large, high-quality datasets tailored to specific modeling languages, often due to privacy concerns. While LLMs can generate synthetic traces to mitigate this issue, the quality of such data remains critical. Generated traces may suffer from inaccuracies or 'hallucinations' outputs where AI systems generate plausible but incorrect or nonsensical information (Muttillo et al., 2024) deviating from correct modeling practices and compromising the reliability of AI-driven tools. Additionally, many AI-based approaches lack generalizability across different modeling tools and environments. Validation is often limited, with some studies relying on a single case study such as a hydraulic test rig (Moin et al., 2022) which may not represent broader domains like IoT. This highlights the need for more comprehensive evaluations across diverse scenarios and use cases. Finally, the use of LLMs in generative AI raises ethical and privacy concerns, particularly when handling sensitive or proprietary modeling data.

5 THREATS TO VALIDITY

To minimize possible threats to validity, particularly threats to internal and construct validity, we followed well-established guidelines for studies during this research. In SLR, one of the main threats to external validity is that primary studies may not be representative of the state of the art and practice. To mitigate this threat, we targeted four well-known scientific databases. These operation also helped us mitigate threats to construct validity. However, we removed studies that were not written in English, but we don't believe there's a significant risk of excluding relevant studies not written in English since English is the de-facto standard language for scientific research in computer science and software engineer- ing. It is possible that SLR cannot answer all research questions.

6 CONCLUSIONS

In our systematic literature review, we have examined a total of 56 primary studies to explore the use of various AI algorithms in MDA. The aim of our review was to address specific research questions and provide an overview of current MDA that is enabled by AI. From this SLR, we conclude that, in recent years, major players have increasingly used model-based technologies to develop industrial software. AI has also seen significant advances recently, especially in Large Language Models.

This paper offers valuable insights for both practition- ers and researchers examining the current state of the field. Furthermore, our SLR can positively influence the research community and facilitate its transition to Generative AI.

In future works, we plan to propose an intelli- gent MDA framework that automates and enhances the generation of requirement, models and the code generation. Also, We aim to utilize a large datasets to prevent underfitting during the training of the AI models.

ACKNOWLEDGEMENTS

This work was partially supported by the LABEX-TA project MeFoGL: ”Me´thodes formelles pour le Ge´nie Logiciel”

REFERENCES

Alulema, D., Criado, J., Iribarne, L., Ferna´ndez-Garc´ıa,A. J., and Ayala, R. (2023). Si4iot: A methodology based on models and services for the integration of iot systems. *Future Generation Computer Systems*, 143:132–151.

Babaalla, Z., Bouziane, E. M., Jakimi, A., and Oualla, M. (2024a). From text-based system specifications to uml diagrams: A bridge between words and models. In *2024 International Conference on Circuit, Systems and Communication (ICCSC)*, pages 1–6.

Babaalla, Z., Jakimi, A., Oualla, M., Saadane, R., and Chehri, A. (2024b). Towards an automatic extracting uml class diagram from system's textual specification. In *Proceedings of the 7th International Conference on Networking, Intelligent Systems and Security*, pages 1–5. Batchkova, I. and Ivanova, T. (2019). Model-driven devel- opment of agent-based cyber-physical systems. *IFAC- PapersOnLine*, 52(25):258–263.

Benaben, F., Lauras, M., Fertier, A., and Salatge´, N. (2019). Integrating model-driven engineering as the next challenge for artificial intelligence – application to risk and crisis management. In *2019 Winter Simulation Conference (WSC)*, pages 1549–1563.

Bhadra, M., Lopera, D. S., Kunzelmann, R., and Ecker, W. (2024). A model-driven architecture approach to accelerate software code generation. In *2024 7th International Conference on Software and System Engineering (ICoSSE)*, pages 23–30. IEEE.

Binder, C., Cala`, A., Vollmar, J., Neureiter, C., and Lu¨der, A. (2021). Automated model transformation in modeling digital twins of industrial internet-of-things applications utilizing automationml. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–6.

Binder, C., Neureiter, C., and Lu¨der, A. (2022). Towards a domain-specific information architecture enabling the investigation and optimization of flexible production systems by utilizing artificial intelligence. *The International Journal of Advanced Manufacturing Technology*, 123(1):49–81.

Biswas, N., Mondal, A. S., Kusumastuti, A., Saha, S., and Mondal, K. C. (2022). Automated credit assessment framework using etl process and machine learning. *Innovations in Systems and Software Engineering*, pages 1–14.

Brandon, C., Boßelmann, S., Singh, A., Ryan, S., Schieweck, A., Fennell, E., Steffen, B., and Margaria, T. (2024). Cinco de bio: A low-code platform for domain-specific workflows for biomedical imaging research. *BioMedInformatics*, 4(3):1865–1883.

Bruneliere, H., Muttillo, V., Eramo, R., Berardinelli, L., Gómez, A., Bagnato, A., Sadovykh, A., and Cicchetti, A. (2022). Aidoart: Ai-augmented automation for devops, a model-based framework for continuous development in cyber–physical systems. *Microprocessors and Microsystems*, 94:104672.

Chang, W., Wei, R., Zhao, S., Wellings, A., Woodcock, J., and Burns, A. (2020). Development automation of real-time java: Model-driven transformation and synthesis. *ACM Transactions on Embedded Computing Systems (TECS)*, 19(5):1–26.

Dorodnykh, N. O., Yurin, A. Y., and Stolbov, A. B. (2018). Ontology driven development of rule-based expert systems. In *2018 3rd Russian-Pacific Conference on Computer Technology and Applications (RPC)*, pages 1–6.

Eramo, R., Said, B., Oriol, M., Bruneliere, H., and Morales, S. (2024). An architecture for model-based and intelligent automation in devops. *Journal of Systems and Software*, 217:112180.

Houghtaling, M. A., Fiorini, S. R., Fabiano, N., Gonc¸alves, P. J. S., Ulgen, O., Haidegger, T., Carbonera, J. L., Olszewska, J. I., Page, B., Murahwi, Z., and Prestes,

E. (2024). Standardizing an ontology for ethically aligned robotic and autonomous systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 54(3):1791–1804.

Iyenghar, P., Otte, F., and Pulvermueller, E. (2022). Ai-guided model-driven embedded software engineering. In *MODELSWARD*, pages 395–404.

Khalfi, M. F., Tabbiche, M. N., and Adjoudj, R. (2024). From programming-to-modeling-to-prompts smart ubiquitous applications. *Journal of Ambient Intelligence and Smart Environments*, (Preprint):1–39.

Koseler, K., McGraw, K., and Stephan, M. (2019). Realization of a machine learning domain specific modeling language: A baseball analytics case study. In *MODELSWARD*, pages 13–24.

Kouissi, M., Ghouch, N. E., and En-naimi, E. M. (2019). New approach for modeling and developing multi-agent systems based on case based reasoning. In *Proceedings of the 4th International Conference on Smart City Applications*, pages 1–8.

Kulkarni, V., Reddy, S., Barat, S., and Dutta, J. (2023). Toward a symbiotic approach leveraging generative ai for model driven engineering. In *2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 184–193. IEEE.

Lano, K. and Xue, Q. (2023). Code generation by example using symbolic machine learning. *SN Computer Science*, 4(2):170.

Li, X.-S., Tao, X.-P., Song, W., and Dong, K. (2018). Aocml: A domain-specific language for model-driven development of activity-oriented context-aware applications. *Journal of Computer Science and Technology*, 33:900–917.

Liu, H., Shen, M., Zhu, J., Niu, N., Li, G., and Zhang, L. (2020). Deep learning based program generation from requirements text: Are we there yet? *IEEE Transactions on Software Engineering*, 48(4):1268–1289.

Lopes, R., Arau´jo, J., da Silveira, D. S., and Sardinha, A. (2024). A systematic approach to derive conceptual models from bpmn models. In *International Symposium on Business Modeling and Software Design*, pages 83–96. Springer.

López, J. A. H., Ca´novas Izquierdo, J. L., and Cuadrado, J. S. (2022). Modelset: a dataset for machine learning in model-driven engineering. *Software and Systems Modeling*, pages 1–20.

Maass, W. and Storey, V. C. (2021). Pairing conceptual modeling with machine learning. *Data & Knowledge Engineering*, 134:101909.

Meyma, M. M., Laaz, N., and Mbarki, S. (2022). A new model-based approach for migrating health 2.0 to health 3.0 applications. In *International Confer- ence on Advanced Intelligent Systems for Sustainable Development*, pages 673–682. Springer.

Moin, A., Challenger, M., Badii, A., and Gu¨nnemann, S. (2021). Mde4qai: Towards model-driven engineer- ing for quantum artificial intelligence. *arXiv preprint arXiv:2107.06708*.

Moin, A., Challenger, M., Badii, A., and Gu¨nnemann, S. (2022). Supporting ai engineering on the iot edgethrough model-driven tinyml. In *2022 IEEE 46th*

An- nual Computers, Software, and Applications Confer- ence (COMPSAC), pages 884–893.

Muttillo, V., Di Sipio, C., Rubei, R., Berardinelli, L., and Dehghani, M. (2024). Towards synthetic trace generation of modeling operations using in-context learning approach. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, pages 619–630.

Mythily, M., Valarmathi, M., and Durai, C. A. D. (2019). Model transformation using logical prediction from sequence diagram: an experimental approach. *Cluster Computing*, 22(Suppl 5):12351–12362.

Naimi, L., Bouziane, E. M., and Jakimi, A. (2024). Automating test case generation from class diagram using generative ai. In *International Conference on Smart Medical, IoT & Artificial Intelligence*, pages 133–140. Springer.

Naveed, H., Grundy, J., Arora, C., Khalajzadeh, H., and Haggag, O. (2024). Towards runtime monitoring for responsible machine learning using model-driven engineering. In *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, pages 195–202.

Ouali, S., Mhiri, M., and Gargouri, F. (2020). Existing business process models for model construction of multidimensional business knowledge: an mda-based transformation methodology. *New Generation Computing*, 38(3):477–508.

Ouchra, H., Belangour, A., Erraissi, A., and Labied, M. (2024). Data to cartography new mde-based approach for urban satellite image classification. *Journal of Environmental & Earth Sciences— Volume*, 7(01).

Panahandeh, M., Hamdaqa, M., Zamani, B., and Hamou-Lhadj, A. (2021). Muppit: A method for using proper patterns in model transformations. *Software and Systems Modeling*, 20(5):1491–1523.

Park, J., Shin, Y., and Sung, H. (2023). Multi-objective architecture search and optimization for heterogeneous neuromorphic architecture. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–8.

Pérez-Castillo, R., Delgado, A., Ruiz, F., Bacigalupe, V., and Piattini, M. (2022). A method for transforming knowledge discovery metamodel to archimate models. *Software And Systems Modeling*, pages 1–26.

Rigou, Y., Lamontagne, D., and Khriss, I. (2020). A sketch of a deep learning approach for discovering uml class diagrams from system's textual specifica- tion. In *2020 1st International Conference on Inno- vative Research in Applied Science, Engineering and Technology (IRASET)*, pages 1–6. IEEE.

Safdar, A., Azam, F., Anwar, M. W., Akram, U., and Rasheed, Y. (2022). Modlf: A model-driven deep learning framework for autonomous vehicle perception (avp). In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems*, pages 187–198.

Sajji, A., Rhazali, Y., and Hadi, Y. (2023). A methodology of automatic class diagrams generation from source code using model-driven architecture and machine learning to achieve energy efficiency. In *E3S Web of Conferences*, volume 412, page 01002. EDP Sciences.

Sarazin, A., Truptil, S., Montarnal, A., Bascans, J., and Lorca, X. (2021). Model transformation from cbm to epl rules to detect failure symptoms. In *Model- Driven Engineering and Software Development: 8th International Conference, MODELSWARD 2020, Valletta, Malta, February 25–27, 2020, Revised Selected Papers 8*, pages 200–224. Springer.

Servadei, L., Zennaro, E., Fritz, T., Devarajegowda, K., Ecker, W., and Wille, R. (2019). Using machine learning for predicting area and firmware metrics of hardware designs from abstract specifications. *Microprocessors and Microsystems*, 71:102853.

Siala, H. A. (2024). Enhancing model-driven reverse engineering using machine learning. In *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*, ICSE-Companion '24, page 173–175, New York, NY, USA. Association for Computing Machinery.

Tabbiche, M. N., Khalfi, M. F., and Adjoudj, R. (2023). Applying machine learning and model-driven approach for the identification and diagnosis of covid-19. *International Journal of Distributed Systems and Technologies (IJDST)*, 14(1):1–27.

Thota, S. R., Arora, S., and Gupta, S. (2024). Al-driven automated software documentation generation for enhanced development productivity. In *2024 International Conference on Data Science and Network Security (ICDSNS)*, pages 1–7.

Tinnes, C., Kehrer, T., Joblin, M., Hohenstein, U., Biesdorf, A., and Apel, S. (2021). Learning domain-specific edit operations from model repositories with frequent subgraph mining. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 930–942.

Uyanık, B. and Sayar, A. (2023). Analysis and comparison of automatic code generation and transformation techniques on low-code platforms. In *Proceedings of the 2023 5th International Conference on Software Engineering and Development*, pages 17–27.

Uzun, B. and Tekinerdogan, B. (2018). Model-driven architecture based testing: A systematic literature review. *Information and Software technology*, 102:30–48.

Yang, S. and Sahraoui, H. (2022). Towards automatically extracting uml class diagrams from natural language specifications. In *Proceedings of the 25th Interna- tional Conference on Model Driven Engineering Lan- guages and Systems: Companion Proceedings*, pages 396–403.

Zennaro, E., Servadei, L., Devarajegowda, K., and Ecker, W. (2018). A machine learning approach for area prediction of hardware designs from abstract specifications. In *2018 21st Euromicro Conference on Digital System Design (DSD)*, pages 413–420.

Zouani, Y. and Lachgar, M. (2024). Zynerator: Bridg- ing model-driven architecture and microservices for enhanced software development. *Electronics*, 13(12):2237.