Model-Driven Engineering for Artificial Intelligence - A Systematic Literature Review*

Simon Rädler[a,b], Luca Berardinelli[c], Karolin Winter[d], Abbas Rahimi[c], Stefanie Rinderle-Ma[a]

[a]*Department of Computer Science, TUM School of Computation, Information and Technology, Technical University of Munich, Boltzmannstr. 3, Garching, 85748, Germany*

[b]*Business Informatics Group, Technical University of Vienna, Favoritenstraße 9-11/194-3, Vienna, 1040, Austria*

[c]*Department of Business Informatics–Software Engineering, Johannes Kepler University, Altenberger Straße 69, Linz, 4040, Austria*

[d]*Department of Industrial Engineering and Innovation Sciences, Eindhoven University of Technology, Groene Loper 3, Eindhoven, 5612 AE, The Netherlands*

Abstract

Background: Technical systems are becoming increasingly complex due to the increasing number of components, functions, and involvement of different disciplines. In this regard, Model-Driven Engineering (MDE) techniques and practices tame complexity during the development process by using models as primary artifacts. Today, the amount of data generated during product development is rapidly growing, leading to an increased need for leveraging Artificial Intelligence (AI) algorithms. However, using these algorithms in practice can be difficult and time-consuming. Therefore, utilizing MDE techniques and tools for formulating AI algorithms or parts of them can reduce these complexities and be advantageous.

Objective: This study aims to investigate the existing body of knowledge

*This project has been partially supported and funded by the AIDOaRt project, an ECSEL Joint Undertaking (JU) under grant agreement No. 101007350 and the Austrian Research Promotion Agency (FFG) via the Austrian Competence Center for Digital Production (CDP) under the contract number 881843.

in the field of MDE in support of AI (MDE4AI) to sharpen future research further and define the current state of the art.

Method: We conducted a Systemic Literature Review (SLR), collecting papers from five major databases resulting in 703 candidate studies, eventually retaining 15 primary studies. Each primary study will be evaluated and discussed with respect to the adoption of (1) MDE principles and practices and (2) the phases of AI development support aligned with the stages of the CRISP-DM methodology.

Results: The study's findings show that the pillar concepts of MDE (metamodel, concrete syntax and model transformation), are leveraged to define domain-specific languages (DSL) explicitly addressing AI concerns. Different MDE technologies are used, leveraging different language workbenches. The most prominent AI-related concerns are training and modeling of the AI algorithm, while minor emphasis is given to the time-consuming preparation of the data sets. Early project phases that support interdisciplinary communication of requirements, such as the CRISP-DM *Business Understanding* phase, are rarely reflected.

Conclusion: The study found that the use of MDE for AI is still in its early stages, and there is no single tool or method that is widely used. Additionally, current approaches tend to focus on specific stages of development rather than providing support for the entire development process. As a result, the study suggests several research directions to further improve the use of MDE for AI and to guide future research in this area.

Keywords: Model-Driven Engineering, Artificial Intelligence, MDE4AI, Domain Specific Language, SLR, Literature Review, Machine Learning
PACS: 0000, 1111
2000 MSC: 0000, 1111

---

1. Introduction

Engineering systems are becoming more complex due to the increasing number of components, functions, and the involvement of several disciplines [5]. To address this complexity, the integration of Model Driven Engineering (MDE) is promising [31, 32, 41].

MDE aims to support the engineering of systems by providing and maintaining information for the development using models rather than documents [31]. Models are enriched with information, shared among stakeholders, and manipulated by Computer-Aided Software Engineering (CASE) tools, aiming at the highest possible degree of automation, e.g., via model transformations. Consequently, MDE techniques allow informed decisions by producing and consuming models as machine-processable artifacts.

Although the models are enriched with information, MDE techniques lack the means to gather sufficient knowledge from more extensive data, e.g., Big Data or data collected from complex (software) systems. In this respect, means of artificial intelligence (AI) and its sub-disciplines, namely machine learning (ML) and deep learning (DL), are beneficial to exploit the information hidden in data [49]. Integrating data-driven methods to support engineering tasks has recently been defined as data-driven engineering [61]. It has proven to be beneficial in several engineering areas such as manufacturing [20], aerospace industry [12] or other industrial applications [6, 23, 57].

AI integration is mainly case-specific, and thus there are several methods supporting the implementation of AI in literature [3], e.g., Cross Industry Standard Process for Data Mining (CRISP-DM) [64], which support the development of AI tools by providing support for the specific development steps typically applied to AI projects. The incorporation of methods such as CRISP-DM and MDE capabilities is rarely considered in the literature, even though the need for experts to implement data-driven solutions is increasing due to the requirement to integrate AI into existing methods [18] and the implementation effort is decreasing by applying MDE principles and practices to the development of AI capabilities.

Recently, a series of AI and MDE workshops was initiated, focusing on using MDE techniques for defining AI methods (MDE4AI) and AI support for MDE (AI4MDE) [16, 14, 13]. The adoption of MDE practices to support AI capabilities of the system under study promises to support the development through degrees of automation of the engineering activities (e.g., code generation), and, therefore, increase the number of industrial applications. Still, to the best of our knowledge, the current state of practice and state of the art is not elaborated regarding MDE approaches that support the implementation of AI capabilities.

In this respect, the overall Research Goal (RG) of this SLR can be defined as

given in Table 1. The RG definition is aligned to the Goal-Question-Metric perspective [4].

Purpose	Collection and comparison of studies on
Issue	model-driven approaches that explicitly address the engineering of
Object	artificial intelligence applications (machine learning, deep learning)
Viewpoint	from the point of view of researchers.

Table 1: The overall research goal.

We define the following overarching Research Question (RQ) based on this research goal:

Main RQ What is the current state of the art for model-driven engineering with extensions to formalize artificial intelligence methods and applications?

To address the main research question, various refined and more fine-grained RQs are introduced in Section 3.

According to the guidelines set out by [36], a systematic literature review (SLR) is conducted to gather and assess existing literature to address the identified research questions [46]. Particularly, this work focuses on the state of the art for MDE approaches that enable the formalization of AI use cases.

The contributions of this SLR comprise:

- Collection and analysis of state of the art model-driven approaches for AI applications.
- Introduction of quantitative assessment criteria for model-driven approaches for AI.
- Quantitative assessment of existing approaches with derived research opportunities.

The remainder of the paper is organized as follows. Section 2 provides background on terms such as MDE and AI. Section 3 introduces the research methodology, i.e., the paper search and selection process. Section 4 presents the approaches aligned with the data extraction strategy of the SLR protocol in Section 3. Section 5 answers the research questions, discusses the key findings, and depicts implications and future research. Section 6 assesses the

quality and limitations of the current SLR using threats to validity analysis. Finally, Section 7 summarizes the paper.

2. Background

This section presents relevant background on MDE and AI. The focus is on fundamentals that support understanding of the SLR results and is not intended to reflect the current state of the art in each research area.

2.1. Model-Driven Engineering

The core of MDE includes the pillar concepts of *model*, *metamodel*, and *model transformation* [8, 53].

Models are machine-readable artifacts representing particular concerns of a system under study, such as design-time information like software architecture or hardware platform, or operational information like monitored data. *Metamodels* define the modeling concepts and their relationships, providing an intentional description of all possible models that must conform to the associated metamodel. From a language engineering perspective, a metamodel represents the *abstract syntax* of a modeling language. *Metamodels* define modeling languages conceptually and are independent of any concrete representation. The *concrete syntax* of a language assigns graphical or textual elements to metamodel elements that can be understood by users and edited through model editors [8, 53]. As models in MDE are considered machine-readable artifacts, so-called *model transformations* apply to modifying existing or generating new engineering artifacts. These artifacts then are used for particular purposes, realizing the steps of the envisioned engineering process toward the (partial or full) generation of the software system.

Depending on the specific engineering concerns (software, hardware, or system as a whole) as well as the role played by model artifacts due to degrees of automation of model management activities (e.g., model-based being to refer to a lighter version of model-driven), different modeling acronyms are typically used (e.g., MBE, MBSE, MDE, MDSE. MDD, MDA)[1]. The various modeling acronyms show that the MDE community is widespread, and the same applies to the goals and applications of MDE approaches.

[1]See `https://modeling-languages.com/clarifying-concepts-mbe-vs-mde-vs-mdd-vs-mda/` for a discussion.

2.2. Artificial Intelligence

Artificial intelligence is a flourishing science with numerous practical applications, ranging from image/voice recognition to recommendation systems and self-driving cars. The primary goal of AI is to tackle problems that are tough for humans but relatively simple for computers [27]. Nevertheless, the different terms around AI are quite fuzzy, and depending on the application area, different terms like machine learning (ML), deep learning, data science, data mining, and so on are used [21]. For example, *data science* is the umbrella term referring to the broad field of extracting information and knowledge by analyzing data to derive patterns, trends, etc., and report them as human-understandable insights [56] that are beneficial for various areas [59], such as manufacturing [20].

Although each term refers to a specific subcategory of data science, the implementation phases are essentially similar. Therefore, in the following, all synonyms and sub-terms will be related to the term AI for better understanding. Consequently, methodologies have been developed to structure and support the implementation of AI projects [64, 3, 22, 24, 66]. The implementation phases of the methodologies in literature are quite similar, although the naming is different [3]. In this work, the focus is on the phases of the CRISP-DM methodology [64]. The main reason for orienting to CRISP-DM is that it is described in the literature as a de-facto standard in the industry and is widely used due to its generality [58, 55]. Additionally, the phases of CRISP-DM can be applied to other sub-topics of data science projects that are not covered under the term data mining. CRISP-DM comprises six phases [64]:

1. The *business understanding* phase involves gathering knowledge on the application domain and the project objectives.
2. The *data understanding* starts with initial data collection and analysis to get familiar with the data.
3. In the *data preparation* phase, the input datasets are built from the given data by applying techniques such as normalization to transform the data.
4. The *modeling phase* applies the AI algorithm to the prepared data, and additional fine-tuning is applied, e.g., hyper-parameter optimization.

5. The *evaluation phase* is used to obtain whether the elaborated model performs as it should.

6. The *deployment phase* deals with infrastructure and the presentation of customer-usable knowledge.

2.3. Related Work

In [48], Portugal et al. surveyed domain-specific languages (DSL) and frameworks for designing ML algorithms for Big Data. The DSLs are described according to the classification in [25, 17, 62]. However, the survey is not systematic, i.e., it does not follow any explicit surveying protocol. Moreover, we propose a classification following DSL engineering principles and practices used in MDE [8, 15]. Moreover, Portugal et al. do not consider DLSs in relationship with the implementation phases of AI algorithms. However, we consider the referenced literature of Portual et al. in our survey for snowballing.

3. Research Methodology

This section introduces the SLR method applied in this work. The SLR study protocol is based on the guidelines by [46, 35, 36], introducing the main steps of SLRs to be performed in the Software Engineering domain.

Figure 1 depicts an activity-like diagram of the performed search and selection process protocol workflow. The workflow consists of the following steps:

1. *Identifying the Research Goals and the Research Questions* (RG/RQ): The objective of this work and the research questions are defined to guide the SLR (Section 1 shows the result of the RG/RQ elaboration)

2. *Search Process*: The literature search is conducted on selected databases collecting scientific publications via the execution of queries based on a search string suitably designed according to the given RGs and RQs (Section 3.2).

3. *Study Selection*: The authors define the inclusion and exclusion criteria (IC/EC) and apply them to the papers collected in the databases by reading their titles and abstracts. Subsequently, the selected papers are evaluated based on their content (Section 3.3).

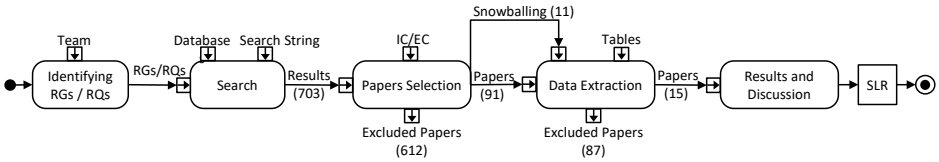

Figure 1: SLR Methodology Overview

4. *Data extraction*: Given a set of selected studies that passed the IC/EC criteria application, detailed data are extracted throughout a full-text reading. In the SLR, papers' detailed information is collected in evaluation tables. If a publication is relevant, snowballing is applied to add referenced papers or the one citing the selected publication (see Section 3.4).

5. *Results Analysis and Discussion*: Collected results are analyzed, and a discussion occurs among the authors to answer the stated RQs.

The execution of the protocol is documented in a spreadsheet[2], and bibliographic entries are collected in Zotero Library. An export can be found online[2].

3.1. Research Questions

The overall research goal is already introduced in Table 1 aligned with the main research question. To answer the main research question, various refined and more fine-grained RQs are defined as follows:

RQ1 What MDE aspects are addressed in the approaches, e.g., abstract syntax (metamodel), concrete syntax, *executable semantics* etc.?

This RQ aims to assess the pillar concepts of MDE languages concerning comprehensiveness (of modeling) and applicability (maturity).

RQ2 Which phases of AI development aligned with the CRISP-DM methodology are covered by the approaches?

This RQ assesses the extent to which the development phases of CRISP-DM are covered. As a result, implications can be made about the extent of support.

[2] `https://github.com/sraedler/Model-Driven-Engineering4Artificial-Intelligence`

RQ3 Which industrial domains are supported by MDE for AI approaches?

This RQ enables finding industries that are using MDE in the context of AI and thus driving the development of MDE for AI using domain-specific tools and methodologies towards the needs of the (specific) industry.

RQ4 What are the used methods and the supporting MDE tools the proposed approaches rely on?

This RQ allows assessing the underlying methods and the related tool support, including further development leveraging these underlying technologies to gain maturity.

RQ5 To what extent is communication between different stakeholders supported by MDE?

Communication and business knowledge elaboration are two of the core pitfalls in the development of AI solutions [47]. Therefore, this question aims to assess the contribution to support fostering AI in the industry.

RQ6 Which challenges and research directions are still open?

This RQ will lead to future research directions and challenges for the model-based engineering of AI applications due to a collection of limitations in the proposed approaches based on respective authors or our obtainment.

3.2. Search Process

This section describes the search activity in Figure 1. According to [36], defined search queries are executed on dedicated search engines. In this research, the queries are performed on the following bibliographic sources:

- ACM Digital Library: `http://dl.acm.org/`
- dblp Computer Science Bibliography: `https://dblp.org/`
- IEEE Xplore Digital Library: `http://ieeexplore.ieee.org`
- Google Scholar: `https://scholar.google.com`
- Springer: `https://link.springer.com`

To select suitable terms for the search, keywords from known studies, the MDE4AI workshop[3] and the International Journal on Software and Systems Modeling (SoSyM)[4] were selected.

The selected keywords for the search terms are the following:

$S_1(MDE) = \{MDE; Model{-}Driven\ Engineering; DSL; DomainSpecific\ Language; Metamodeling; Domain\ Modeling\}$
$S_2(AI) = \{AI; Artificial\ Intelligence; ML; Machine\ Learning; Deep\ Learning; Intelligence\}$

Each keyword k_i from the set S_1 and S_2 has been combined in conjunctive logic proposition $p \in P$.

$$P = \{p | p = s_i \in S_1 \wedge s_j \in S_2\}$$

$$i = 1, 2, 3, 4, 5, 6,\ j = 1, 2, 3, 4, 5, 6$$

The resulting set P of 36 propositions (p_i) includes the final *search strings*. According to [36], the propositions (p_i) should be combined as OR statements. However, for some search engines, a single search term is too complicated, as some search engines limit the length of the search term or do not generate results correctly due to nested search terms. Therefore, each search string is executed as a single query.

The automated search was executed in November 2022. In total, 703 papers have been collected. The search terms and results are archived and are online available[2]. If a result file is unavailable, the search query on the specific search engine did not retrieve any results.

3.3. Paper Selection

The inclusion and exclusion criteria (IC/EC) as outlined in Tab. 2 are employed for the paper selection. The IC/EC have been evaluated for each paper collected by queries executed on the selected databases by reading its title and abstract. Additionally, doctoral theses are excluded due to the extensiveness, but the referenced publications of the author are included in snowballing. Although review papers are not considered for the survey, we presented relevant surveys in the background section (Section. 2.3).

[3] `https://mde-intelligence.github.io/`
[4] `https://www.sosym.org/`

Table 2: Inclusion Criteria (IC) and Exclusion Criteria (EC)

Type	ID	Type
IC	1	We include system-level DSL (metamodel) with AI extensions
	2	We include data-driven/model-driven approaches with AI extensions
EC	1	We exclude simulation-based (only) approaches
	2	We exclude algorithm-based (only) approaches
	3	We exclude secondary studies
	4	We exclude review papers, but include them in snowballing
	5	We exclude study available only in form of abstract
	6	We exclude study not in English language
	7	We exclude papers with focus on software architecture for MDE for AI, e.g. Hadoop integration in infrastructure
	8	We exclude vision only papers and proposals

Following the IC/EC criteria application, a full-paper read is applied to select the final papers. Additionally, snowballing is accomplished as suggested by [36] to retrieve further results. The relevant papers from the list of snowballing papers were selected with the same procedure as the query results. Table 3 lists the final list of selected papers. Particularly, 11 papers are added by query selection, and four are added due to snowballing. The strong selection of results from the heavy use of AI/ML as keywords in the publications and the actuality of the research approach, see Workshop on Modeling Intelligence [14, 16].

3.4. Data Extraction

Each selected paper presented in Table 3 underwent a data extraction process following the data extraction template in Table 4. Additionally, the publication type is assessed as Exploratory (without evaluation, e.g., a pure concept or vision) or Technical (with evaluation).

Table 3: List of selected publications with type of publication incl. snowballing results.

Item Type	**Year**	**Author**	**Title**
Conference Paper	2020	[2]	Model Driven Approach for Neural Networks
Conference Paper	2019	[7]	STRATUM: A BigData-as-a-Service for Lifecycle Management of IoT Analytics Applications
Journal Article	2020	[19]	Lavoisier: A DSL for increasing the level of abstraction of data selection and formatting in data mining
Journal Article	2022	[26]	A domain-specific language for describing machine learning dataset
Conference Paper	2017	[28]	The next Evolution of MDE: A Seamless Integration of Machine Learning into Domain Modeling
Conference Paper	2019	[29]	Meta-Modelling Meta-Learning
Conference Paper	2019	[30]	Model-based design for CPS with learning-enabled components
Conference Paper	2019	[37]	Realization of a Machine Learning Domain Specific Modeling Language: A Baseball Analytics Case Study
Conference Paper	2019	[38]	On the Engineering of AI-Powered Systems
Journal Article	2021	[42]	AdaptiveSystems: An Integrated Framework for Adaptive Systems Design and Development Using MPS JetBrains Domain-Specific Modeling Environment
Conference Paper	2022	[43]	A Model-Driven Approach for Systematic Reproducibility and Replicability of Data Science Projects
Journal Article	2021	[44]	A Model-Driven Engineering Approach to Machine Learning and Software Modeling
Journal Article	2022	[45]	Towards a DSL for AI Engineering Process Modeling
Conference Paper	2021	[51]	An MDE Method for Improving Deep Learning Dataset Requirements Engineering using Alloy and UML
Conference Paper	2020	[67]	Arbiter: A Domain-Specific Language for Ethical Machine Learning

Table 4: Data Extraction Template

RQ	**Concern**		**Assessment Description**
RQ1	MDE	Metamodels	The metamodel of the approach is either depicted as a diagram in a referenced repository or clearly mentioned and textually described.
		Concrete Syntax	The concrete syntax is given if figures, listings, or tables to illustrate an implementation/use case excerpt or it is indicated whether textual or graphical modeling is applied for a specific aspect.
		Arbitrary Constraints	The approach or the underlying modeling framework (e.g., SysML) allows the specification of arbitrary constraints.
		Model Transformation	The approach uses or introduces model transformation to generate engineering artifacts of any kind.
RQ2, RQ5	AI	Business Understanding	The model contributes to the understanding of the underlying business. Particularly, the creation of the data and aspects from other disciplines are introduced, such as requirements modeling for AI.
		Data Understanding	The model supports at least two of the following aspects: data description, data attribute interrelation, data background, data quality, and data composition.
		Data Ingestion	The model clearly depicts the origin of data and how to load it.
		Feature Preparation	The model allows an understanding of how data needs to be transformed, connected, or preprocessed.
		Model Training	The model depicts the used algorithm with input and output values and potential hyperparameters.
		Metrics/Evaluation	The model depicts metrics for the AI approach or introduces evaluation criteria.
RQ3	Others	Problem Domain	The domain of the case study or the mentioned area of application.
RQ4		Frameworks	The method and tools used in the approach, e.g., WebGME, Xtext, Xtend, etc.

The extracted data mainly address two concerns of interest, i.e., MDE and AI. Modeling concerns refer to the evidence of sound knowledge and application of *model foundations* [15] (e.g., abstract syntax/grammar/metamodel, textual/graphical concrete syntax, constraints, model transformations) and supporting tools (e.g., modeling language frameworks). AI concerns [1] indicate to which extent the publications support ML modeling aligned with the dimensions of the CRISP-DM methodology [64]. It should be noted that the assessment dimensions do not correspond exactly to the phases of CIRSP-DM to allow for a more detailed categorization of concerns; e.g., in CRISP-DM, *Data Ingestion* is part of the *Data Understanding* phase but separated in the given assessment. An aspect of a concern of interest is assessed as *available* (✓) if the aspect is presented in the approach or as *underlying principle* is typically offered by the underlying environment (e.g., constraint modeling might not be presented but is typically offered by the underlying MDE tooling). Finally, it is worth noting that there is no evaluation of the deployment phase of CRISP-DM as it is beyond the scope of this paper.

4. Literature Assessment

The result obtained from the data extraction process described in the previous section is presented in Tables 5, 7 and 8.

Table 5: Result of the data extraction for the MDE and AI concerns.

RQ	Concern	Paper / Criteria	[2]	[7]	[19]	[26]	[28]	[29]	[30]	[37]	[38]	[42]	[43]	[44]	[45]	[51]	[67]	Sum
	General	**T**echnical or **E**xploratory Paper	**T**	**T**	**T**	**T**	**T**	**E**	**T**	**E**	**T**	**T**	**T**	**T**	**E**	**T**	**T**	12 **T** 3 **E**
RQ1	MDE	Metamodels	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		14
		Concrete Syntax	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	14
		Arbitrary Constraints	✓	✓					✓	✓			✓			✓	✓	7
		Model Transformation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓		✓		12
RQ2, RQ5	AI	Business Understanding							✓						✓	✓	✓	4
		Data Understanding			✓	✓	✓	✓		✓			✓			✓		7
		Data Ingestion	✓	✓	✓				✓		✓	✓	✓	✓	✓		✓	10
		Feature Preparation	✓	✓									✓	✓	✓			5
		Model Training	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓			11
		Metrics/Evaluation		✓					✓		✓			✓				4

In [2], Al-Azzoni proposes a model-driven approach to describe ML problems addressed by artificial neural networks. The approach enables the description of datasets as well as the consuming Multi-Layer Perception (MLP) Neural Networks (NN). With templates and code generators, executable Java programs can be generated. The approach is validated using the Pima Indians Diabetes dataset.

In [7], Bhattacharjee et al. introduce STRATUM, a model-driven tool that enables dealing with the lifecycle of intelligent component development. The platform addresses design-related concerns such as modeling the ML algorithm pipeline, accessing data streams, allocating and properly sizing cloud-based execution platforms, and monitoring the overall system's quality of service. The primary goal of this work is to support deploying and maintaining various cloud-based execution platforms. The MDE part of this work is minor and less detailed.

In [19], De La Vega et al. introduce a DSL that describes datasets to select sufficient data on a high level. The approach uses a SQL-like textual language to select, combine and filter various data on an attribute level. The approach aims to increase a dataset's abstraction level to reduce complexity and make using data mining technologies easier.

In [26], the DescribeML DSL is proposed to define ML datasets. From a DescribeML model, a template with basic information is automatically generated, based on a given dataset. The provided DSL allows the definition of metadata, data attributes with statistical features and provenance, and social concerns. This approach aims to improve the understanding of datasets and thus support the replicability of AI projects. Currently, this work is limited to the dataset description. Future work aims to describe AI models and other elements of an AI pipeline.

In [28], Hartmann et al. present an approach based on so-called micro-learning units at a language definition level. This work proposes to weave the learning units into domain modeling, due to the high entanglement of learning units and domain knowledge. For this purpose, the approach allows the definition of DSLs with learned attributes (i.e., what should be learned), how (i.e., algorithm and parameters), and from what (i.e., other attributes and relations).

Hartmann et al. leverage the previous study for meta-learning in [29]. This

study proposes two generic metamodels for modeling i) ML algorithms and ii) meta-ML algorithms (i.e., algorithms to learn ML ones).

In [30], a comprehensive modeling environment for learning-enabled components in CPS development is introduced. The approach supports training, data collection, evaluation, and verification. It integrates Goal Structuring Notation (GSN) to support assurance and safety cases. The publication is, among others, part of a research project[5] facilitating MDE.

In [37], a DSL is introduced with the goal of proving the plausibility of using MDE approaches to create ML software. The DSL, conceptually sketched by another research group in [9], is realized and applied to a case study in the sports domain. The approach integrates model transformation to generate executable code.

In [38], an approach describing deep learning using MDE is presented. The approach combines two DSLs, namely MontiAnna, and EmbeddedMontiArc. The former is a textual modeling framework for designing and training Artificial Neural Networks (ANNs). It also embeds another DSL, MontiAnna-Train, for describing the training procedure. The latter, EmbeddedMontiArc, is an architectural description language. It supports the definition of components and connectors, with a particular focus on embedded, automotive, and cyber-physical systems. The frameworks are intended to define deep artificial neural networks, e.g., convolutional neural networks, for processing traffic images to learn how to drive a car in a simulator.

In [42], Meacham et al. propose a set of DSLs and toolset implemented on top of the MPS[6] language workbench for the design and development of adaptive systems offering MAPE-K and AI in context capabilities. The approach describes an extension and composition of DLSs that are extended with application-specific concepts.

In [43], Melchor et al. propose an MDE approach to formalizing ML projects and the associated infrastructure in which the resulting tool will be deployed. The approach aims to increase the reproducibility and replicability of data science projects. Hence a key feature of the approach is to describe processes and datasets in detail.

[5] `https://modelbasedassurance.org/`

[6] Meta Programming System

In [44], Moin et al. present an MDE approach based on ThingML[7] to support the development of IoT devices with the extension of data analytics and ML. The ThingML framework supports defining software parts and components using UML. The communication between the components (things) is defined using ports, messages, and state machines. The approach supports the transformation of the model into executable code.

In [45], Morales et al. provide a DSL to model AI-related processes using Eclipse-based technologies. The approach aims to describe AI processes within an organization and thus contribute to the structured designing, enacting, and automating of AI engineering processes.

A model-driven engineering approach for defining dataset requirements is introduced in [51]. It focuses on the structural definition of requirements using semi-formal modeling techniques.

In [67], Zucker et al. present a very preliminary version of a declarative DSL for ethical AI addressing transparency, fairness, accountability, and reproducibility concerns of ethical machine-learning datasets. The approach describes datasets in a SQL-fashioned language and provides a notation to record how ML models will be trained.

4.1. Model-Driven Engineering Concerns

In this section, we report the contributions of the selected studies with respect to MDE techniques and practices [8, 15], i.e., metamodels/grammars, graphical/textual concrete syntax, constraints, and model transformations. In particular, we consider whether the proposed approaches leverage language workbenches [33] to create DSLs adopted in the presented approaches.

Abstract Syntax

In 14 out of 15 approaches, the abstract syntax of one or more DSLs is defined by metamodels or grammars. The only exception is [67], where the authors explicitly remark that the proposed DSL is a preliminary ad-hoc implementation for the proposed case study and does not provide any grammar or metamodel specifications.

In the following, we classify the selected studies based on the technologies used to specify the abstract syntax of DSLs used in the proposed ap-

[7] `https://github.com/TelluIoT/ThingML`

proaches [33]. A large majority of the selected studies, i.e., eleven, adopt a metamodel-centric language design [2, 7, 19, 28, 29, 30, 37, 43, 44, 45, 51] by leveraging EMF and WebGME language frameworks, two adopts a grammar-centric approach [26, 38] by leveraging Langium and Monticore language workbenches, and one a projectional [42] approach, based on JetBrains MPS.

EMF-based. In eight studies, the metamodel is based on EMF [2, 19, 26, 37, 43, 44, 45, 51]. Several EMF metamodels focus on the description of datasets [2, 19, 26, 51]. Other studies additionally describe algorithms [2, 37, 43] or even further steps of the implementation [44, 45]. In [37], the conceptual metamodel presented as an entity-relationship diagram in [9] is realized as a UML profile, i.e., a lightweight extension of the UML metamodel in Papyrus UML, which leverages EMF.

KMF/Greycat-based. Two studies [28, 29] of the same research group are based on the Kevoree Modeling Framework (KMF) and its successor GreyCat, which results from a research project to create an alternative to the EMF based on Ecore. In [28], the capabilities of the Greycat metalanguage are presented. In particular, it allows the definition of microlearning units by explicitly declaring *learned attributes* as part of the domain-specific metamodels. In [29], two metamodels for ML and meta-learning are proposed. The former contains definitions for datasets, metadata, and learning algorithm with hyper-parameters.

WebGME-based. Two studies [7, 30] define metamodels using the WebGME metamodeling framework. While UML and profiles cannot provide the language engineering support typically offered by language workbenches, WebGME allows specifying DSLs creating a class diagram-based metamodel from which the DSL infrastructure is automatically generated. In [7], the so-called Stratum approach for BigData-as-a-Service provides a DSML consisting of several metamodels built on top of WebGME (metamodel for ML algorithms, metamodel for data ingestion frameworks, metamodel for data analytics applications, metamodels for heterogeneous resources). In [30], the metamodel is based on existing metamodel libraries: SEAM, DeepForge, and ROSMOD.

Langium-based. In [26], the DescribeML DSL is the only work leveraging the recent Langium open-source language workbench enabling domain-specific

languages in VS Code, Eclipse Theia, and web applications, leveraging the Language Service Protocol (LSP)[8]. In [26], three metamodels are described i) metadata model, ii) composition model, and iii) provenance and social concerns model. Such metamodels are then implemented as grammars[9].

MontiCore-based. In [38], all DSLs, i.e., MontiAnna, MontiAnnaTrain, and EmbeddedMontiArc, are all defined using the MontiCore language workbench [54]. One of the main benefit is the reuse of existing C++ code generators for neural network frameworks (MxNet, Caffe2, and Tensorflow).

MPS-based. In [42], five different DSLs are created with JetBrains MPS, an open-source projectional language workbench that allows direct changes to the abstract syntax tree through an editor, without the need for a grammar or parser. [42] leverages MPS' language extension and composition capabilities to deal with domain-independent (e.g., using the AdaptiveSystems DSL to structure the system according to MAPE-K loop by IBM) and domain-specific concerns (e.g., AdaptiveVLE to model concerns of virtual learning environments).

Concrete Syntax

This section assesses the proposed approaches' notations or *concrete syntax*. A concrete syntax is explicitly mentioned by 13 out of 15 approaches.

Seven studies [19, 26, 28, 29, 42, 44, 67] provide a textual (or tabular) notation; five studies [7, 30, 37, 45, 51] adopt a graphical notation; one [38] offers both a textual and a graphical notation.

No concrete syntax available. Two studies [2, 43] do not provide a DSL-specific concrete notation. In particular, Al-azzoni [2] left the definition of a complete DSL as future work while [43] is conceived to reuse the notations offered by tools defining data science pipelines. However, by leveraging EMF, a tree-based notation is possible by automatically generated editors, and, potentially, compatible technologies can provide textual or graphical concrete syntax options (e.g., via Xtext and Sirius, respectively).

[8] `https://microsoft.github.io/language-server-protocol/specifications/lsp/3.17/specification/`

[9] Based on Chevrotain, `https://chevrotain.io/docs/`.

Textual notation. In [19], De La Vega et al. provide a textual concrete syntax for the Lavoisier DSL defined by an Xtext-based grammar. Similarly, in [44], the approach is built on top of ThingML and, as such, it provides an Xtext-based textual editor. In [26], the textual concrete syntax is defined by a recent language workbench, Langium. In [28] and [29], an Emfatic-inspired textual modeling language is defined. In [42], five different interwoven DSLs, are proposed, mixing textual and tabular projections, created with JetBrains MPS. In [67], a SQL-like textual notation is proposed. However, they do not provide any grammar, and then the textual notation is just a proposal.

Graphical notation. In [7] and [30], the graphical concrete syntax is defined through capabilities offered by the WebGME language framework. [37] implements the metamodel as a UML profile in Papyrus. The UML Class Diagram is chosen as graphical notation since all the stereotypes inherit from the Class metaclass. No DSL-specific customization of the UML graphical notation is offered. [45] provides a web-based graphical editor realized using Sirius Web[10]. In [51], the DSL provides a graphical concrete syntax and editor realized in Sirius[11]. However, the paper does not discuss or show its graphical elements.

Multiple notation. In [38], Kushmenko et al. are the only ones proposing a mix of textual and graphical concrete notations to represent AI concerns. However, it is worth noting that the SVG-based hierarchical representation of components and connectors is made for visualization purposes and is not editable[12].

Model Transformation

Twelve selected studies include model transformations as part of the proposed approaches. These model transformations are classified based on their intents, as described in [40], and the technology they use, as described in [34]. Table 6 summarizes the intents of the model transformation for each paper, as well as the main model-driven technologies used. It is important to note that none of the papers explicitly list or classify their model transformations. The identification of existing transformations and their intents is an attempt by the authors of this paper to provide a basis for comparison.

[10] `https://www.eclipse.org/sirius/sirius-web.html`

[11] `https://www.eclipse.org/sirius`

[12] `https://github.com/EmbeddedMontiArc/Documentation`

Table 6: Model transformation intent category and concrete intent.

Paper	**Intent Category**	**Concrete Intent**	**Tool**
[2]	Refinement	Model to Code	Epsilon EGL
[7]	Refinement	Model to Code	JS Implementation
[19]	Language Translation	Translation	Xtend
	Abstraction	Restrictive Query	
[26]	Language Translation	Translation	Typescript (Visual Studio Code)
[29]	Refinement	Refinement	n.a.
	Refinement	Model to Code	
[28]	Refinement	Model to Code	KMF/GreyCat
	Semantic Definition	Translational Semantics	
[30]	Refinement	Model to Code	n.a.
	Semantic Definition	Translational Semantics	
	Analysis	Safety Analysis (added)	
[37]	Refinement	Model to Code	Epsilon EGX/EGL/EOL
[38]	Refinement	Model to Code	EmbeddeMontiArc /EMADL2CPP
[42]	Refinement	Model to Code	Jetbrains MPS
	Model Composition	Model Merging	
[44]	Refinement	Model to Code	Xtend
[51]	Semantic Definition	Translational Semantics	Xtend

Nine studies leverage model-to-code transformations [2, 7, 28, 29, 30, 37, 38, 42, 44] to perform refinements on involved artifacts to generate executable code. Three studies [28, 30, 51] aim at executable models by defining translational semantics for their DSLs. Five approaches [19, 29, 28, 30, 42] provide more than one transformation with different intents. Two approaches [19, 26] translate artifacts across different modeling languages

The rightmost column in Table 6 mentions the main model-driven technology leveraged by the studies to implement model transformations.

The most commonly used platform among the studies is Eclipse, with Epsilon[13] and Xtend[14] being the most popular tools. For example, in [2], the Epsilon Generation Language (EGL) is used in conjunction with templates to define model transformations that generate Java code. Similarly, [37] uses EGL to generate C# code for making predictions on test data. In [7], WebGME's code generation capabilities are extended with templates for each sub-task. In [19], two intents of model transformations are reflected: language translation and abstraction using a restrictive query. The model transformation, based on Xtend, transforms dataset descriptions into tabular datasets using low-level data transformation operations, which can then be used in data mining algorithms. In [28], the GreyCat framework, built on the KMF, provides code generation toolsets for building object-oriented applications. In [29], the concept of using code generators to generate ML code is mentioned. In [30], the ALC toolchain enables code generation for data collection or training exercises of learning-enabled components, as well as translational semantics for configuring an embedded Jupyter Notebook that executes the learning model. The approach also allows for the construction of safety cases. In [38], the MontiAnna2X code generator generates MxNet, Caffe2, or Tensorflow code. In [42], JetBrains MPS language is used to generate Java code. In [44], Java and Xtend are used to generate Python code. Finally, in [51], model-to-code transformation is used to complete formal specifications using the Alloy Analyzer.

4.2. Artificial Intelligence Concerns

Same as for the MDE concerns, the findings regarding AI development characteristics are presented in the following.

[13] `https://www.eclipse.org/epsilon`
[14] `https://www.eclipse.org/xtend`

Business Understanding

Industry often faces the problem of missing business understanding and shortcomings in elaborating business values [50, 10, 11, 60]. Therefore, modeling business understanding is essential for mature and comprehensive approaches, e.g., by defining requirements. The assessment revealed that four of the 15 approaches foster business understanding by integrating system-relevant modeling or processes.

In [30], the business understanding is fostered due to requirements and components modeling using SysML. Particularly, a Goal Structuring Notation (GSN) approach is used to define and structure requirements.

In [45], business-relevant information is modeled through the integration of *Roles*, leading to increased business understanding. Additionally, the metamodel reflects means to model requirements. However, details are currently missing on how the modeling is defined.

In [67], requirements on ethical ML can be formalized. Particularly, transparency, accountability and fairness are taken into account so that specific attributes are protected during the implementation, e.g., attributes consisting of values such as 'race' or 'age'.

In [51], a method to describe ML datasets from an requirements engineering perspective is presented. Notably, functional and non-functional requirements are integrated to describe dataset structural requirements.

Data Understanding

The data understanding fosters the downstream processes of CRISP-DM. Additionally, it allows assessing dataset quality and streamlining to form hypotheses for hidden information [64]. In the selected literature, seven approaches support modeling some aspects of the data understanding.

[19] contextualizes dataset properties and improves data understanding by implicitly applying rules on how to select data. In [26], a detailed description of a dataset and data composition is given that fosters the overall data understanding. In [37], data understanding is enhanced due to the input data's graphical representation and the variables' composition. In [43], data understanding is promoted by describing data attributes such as the data type. Furthermore, the type of ML algorithm is described, allowing the reproduction of an ML project.

In [28, 29], the enrichment of properties on a metamodel-level is enabled, which contributes to further description of the properties and, therefore, increases data understanding. Moreover, the interconnection of the data properties is highlighted by the underlying principle. Still, the description of the attributes is not very detailed, leading to no support in understanding a single property and its origin. In [51], the advanced requirements modeling allows for understanding datasets with specific properties and structured data elements better.

Data Ingestion

Ten of the given 15 approaches describe the loading and ingestion of data. Data ingestion, in this sense, refers to the loading or referencing of the input datasets.

In [42], the implementation of data ingestion using a DSL is described. Six other approaches support the specification of a file path, URI, URL, etc., to reference data [43, 67, 30, 2, 19, 45]. In [19], the loading of the dataset is described by specifying the name and path of the file or SQL server in combination with SQL selection scripts. Therefore, this approach supports both file and database-related data. In [45], data loading from various sources, such as SQL servers, is supported.

In contrast, to fix data sources, the loading from edge devices or sensors is supported by three approaches [7, 38, 44]. In [7], data loading from various edge devices is presented using technologies such as RabittMQ or Kafka. In [38], data loading is provided with tagging schemas for EMADL ports. In [44], two approaches are given, first a black-box approach, where the ML model is imported from a pickle, and second, the paths or URLs of the dataset(s) are passed to the training, validation, and testing of the algorithm.

Feature Preparation

The preparation of features for certain ML algorithms is supported by five of the 15 approaches.

In [7], the feature preparation is defined in the metamodel. Unfortunately, details on the specific methods, parameters, or the order of execution are missing. In [2], normalization of dataset features is supported. However, other pre-processing methods are not supported in the metamodel. In [43],

data operations contain one or more input or output ports. Each data operation is an atomic operation on the input data to produce certain output data. In [44], each state allows executing functions. The keyword *DA_Preprocess* is used to apply data preparation methods on a specific dataset. In [45], features can be prepared with specific *feature extraction techniques*, and data can be transformed with data engineering techniques, e.g., *Regression substitution*.

Model Training

The specification of an algorithm and the related training of the model is depicted in 11 of the 15 approaches. The types of algorithms can be separated in Inference [37], Machine Learning [28, 29, 42, 43, 45] and Deep Learning [2, 7, 30, 38, 44] using Neural Networks.

Inference. [37] extended the approach of [9] with the required implementation using SysML and Papyrus modeling framework. Within the original approach [9], model training is given by an assignment for each variable, whether it is an observed variable, a random variable, or a standard variable. Details on hyper-parameter tuning are not given.

Machine Learning. In [28, 29], various algorithm models can be used with specific input (learning) and output attributes. In [42], the algorithm (referred to as *approach*) is specified aligned with various hyper-parameters, e.g., Random Forest Cross Validation Folds. In [43], the algorithm type (e.g., Random Forest) with a specific task type (e.g., Classification) can be described. Hyper-parameters are not presented in the metamodel. In [45], hyper-parameters and performance criteria can be specified for each AI model.

Deep Learning. In [2], the training is defined using an *MLPDescription* block with certain learning rules like BackpropagationFurther details on other hyper-parameters or the output's facilitation are not given. In [7], an algorithm for the training is defined in the metamodel. Moreover, hyper-parameters are defined and applied to a specific algorithm in the editor. In [30], an experimental model defines the model training. The details of the implementation can be found in the Jupyter Notebooks. In [38], the training of NN is given with possibilities to specify the network layers and connections. In [44], state diagrams are used to define various steps of the algorithm. With the state keyword *DA_Train*, various training-related set-

tings are made, and with *DA_Predict*, the trained model can be applied to data.

Metrics/Evaluation
To assess the validity of an algorithm, four of the 15 approaches integrate the modeling of metrics.

In [30], the metrics are applied directly in the Jupyter Notebooks, which is not actually a modeling approach. Nevertheless, the Jupyter Notebook is integrated into the model. So it can be considered as part of the model.

In [7], metrics are integrated into the metamodel and can be applied to the training output. In [38], the evaluation metrics are selected using the name of the metrics, e.g., Mean Squared Error (MSE).

In [44], basic metrics such as Mean Absolute Error (MAE) or MSE can be applied to the algorithms, such as regression algorithms.

4.3. Frameworks (Methods & Tools)
Most of the approaches are based on frameworks and tools. Table 7 depicts each approach's used frameworks and tools. Most of the approaches do not particularly mention the underlying methods. Still, similarities can be seen.

The approach of [9] presents a concept and not an implementation, which is why no tools and underlying methods are depicted.

4.4. Artifacts Available and Domain of Application
Artifacts are a means to enable the replication of research results. Table 8 shows whether artifacts are depicted in the publication as a reference to an online resource or not present. Additionally, the type of application mentioned in the publication or inherently given through the evaluation sample is depicted in the table. If no specific domain is mentioned or derivable, *Unknown* is annotated.

As a result, eight approaches work with datasets, which can originate from any domain. The processing of IoT data is presented in five approaches, whereas one is more specific for image data.

5. Results and Discussion

The discussion is organized according to the research questions in Section 3.1.

Table 7: Used Methods and Tools (RQ4)

Tool or Method \ Paper	[2]	[7]	[19]	[26]	[28]	[29]	[30]	[37]	[38]	[42]	[43]	[44]	[45]	[51]	[67]
Alloy														✓	
BPMN														✓	
DeepForge							✓								
EMF-based	✓		✓								✓			✓	
Epsilon	✓							✓							
GSN							✓								
GreyCat					✓	✓									
Jetbrains MPS										✓					
Jupyter Notebook							✓							✓	
Langium				✓											
MontiCore Workbench									✓						
Papyrus								✓				✓			
Python			✓												✓
ROSMOD							✓								
SEAM							✓								
SQL														✓	✓
Sirius						✓							✓		
ThingML												✓			
WebGME		✓					✓								
Xtext			✓									✓		✓	

Table 8: Availability and type of artifacts aligned with the type of application.

Publication	**In the Publication**	**Online (Git, Server)**	**No Artifacts**	**Type of Application**
[2]	✓			Datasets
[7]		✓		IoT
[19]		✓		Datasets
[26]		✓		Datasets
[28]		✓		IoT
[29]		✓		Unknown
[30]		✓		IoT (CPS)
[37]		✓		Datasets
[38]		✓		IoT (Image)
[42]			✓	Adaptive Systems
[43]		✓		Datasets
[44]		✓		IoT
[45]	✓			Unknown
[51]		✓		Datasets
[67]		✓		Datasets

5.1. RQ1 - What language aspects of MDE are addressed in the approaches, e.g., abstract syntax, concrete syntax, metamodel, etc.?

From a language engineering perspective, each dimension is reflected in most approaches. As for concrete syntax, sometimes an example with concrete syntax is given, but the whole definition of the syntax is not presented.

The description of constraints is rarely used. A reason might be that constraints are often rule-based terms, which can be eliminated with specific parameters or algorithms from the AI domain.

Although not all approaches define a model transformation, most generate artifacts from the models, like the execution code using Python or Jupyter Notebooks.

The review revealed that current approaches are quite diverse from a language technology perspective. In addition, most approaches rely on textual rather than graphical modeling.

5.2. RQ2 - Which phases of AI development aligned with the CRISP-DM methodology are covered by the approaches?

The CRISP-DM development cycle's supported phases are less balanced than the model-driven engineering perspectives. More than half of the approaches support the early phases, such as business understanding. The feature preparation is often not mentioned or integrated with only simple features, e.g., normalization of variables is given but not the subsequent processing of preprocessing tasks. The main focus of the approaches lays in the formalization of model training. However, most of the approaches only support a small range of algorithms. Therefore, the applicability might be very case specific and less flexible.

In summary, it can be seen that multiple approaches depict a specific aspect of the CRISP-DM development cycle, but only a few support more than half of the phases.

5.3. RQ3 - Which industrial domains are supported by MDE for AI approaches?

Most approaches support processing datasets in specific file formats or using data from SQL servers. Since these datasets can originate from any domain, no focus on a domain can be determined in these approaches.

However, some approaches are rather based on IoT/CPS or sensor data, supporting the integration of production systems or data from the use of e.g., CPS products. Nevertheless, no domain can be clearly defined here since collecting sensor data is possible in any domain.

5.4. RQ4 - What are the used methods and MDE tools the proposed approaches rely on?

The present works are based on a wide variety of tools and methods. One reason for this could be the application domain, e.g., SysML would rather be used as a basis if the integration into a mechanical engineering environment is intended since SysML is used anyway. The advantages and disadvantages of the individual methods and tools are therefore considered application-dependent, and no statement can be made about the quality of the underlying methods. Furthermore, there is a trend towards Eclipse and its products (Papyrus, Sirius, Epsilon, etc). The use of EMF for the definition of metamodels or as a basic modeling construct can also be identified as state of the art.

5.5. RQ5 - To what extent is communication between different stakeholders supported by MDE?

Communication in an AI project can be fostered by unifying the language of communication, potentially leading to a better understanding and reduced unknown knowledge among team members. With less unknown knowledge, unrealistic expectations might be reduced, being one of the categories of why AI projects fail [63]. The intersection with other domains is mainly in the initial phases of an AI project, mainly the business, and data understanding. Still, the documentation of other phases of the CRISP-DM cycle supports communication among other AI experts. With respect to interdisciplinary communication, only three approaches support the documentation and integration of business understanding, leading to further research needs. Data understanding and the downstream processes of the CRISP-DM are more often supported. However, still, further integration of MDE techniques is required due to the early development of some of the approaches.

5.6. RQ6 - Which challenges and research directions are still open?

The researchers' observation selected the direction of future research and open challenges. The first observation is that business understanding needs to be more supported. In literature, experts report needing more business

values for AI as a challenge, which potentially originates from the missing understanding of AI experts in the specific business. Consequently, AI experts may not suggest appropriate AI approaches that are realistic and relevant. Aligned with the business understanding, the requirements of a project need to be formalized to allow the derivation of project metrics and further assess the impact of the computational support [52]. Considering that the second largest group of supported applications in the existing works is IoT, perhaps systems engineering approaches should be more integrated into MDE for AI. The definition of requirements or the modeling of the environment could also be borrowed and adapted from these approaches.

Another future work that supports the maturity of MDE for AI is consolidating the advantages of the existing approaches and extending these approaches to fit various use cases. The combination of various approaches to a comprehensive methodology regarding MDE for AI could streamline the research topic and foster the development of MDE for AI toolboxes.

Apart from combining the research workforces, future research needs to focus on the collaboration of engineers using methods designed for concurrently working on models. As of the review's findings, the approaches mainly focus on supporting single editors and do not support collaborative work on a single model. With respect to more extensive or interdisciplinary projects, the live or collaborative work on a single model could increase the development performance and the benefit and acceptance of MDE for AI.

Next, the output of MDE for AI is often a derivation of Python code, etc., based on model transformation. Python is an easy-to-understand, well-known, daily-used language of AI experts that might lead to changes in the Python code rather than the model. Consequently, full code generation is not applied, leading to no single source of information because partial truth of information is stored in the model and partial in the Python code [8]. In this context, it is necessary to elaborate a closed-loop process that feeds the results of the executed algorithm back into the model or adjusts the model in case of changes in the code, e.g., in Python. With this closed-loop approach, the model is always up-to-date, and further, the collaboration with others potentially improves because of the abstracted representation of the actual changes.

Finally, only a few approaches mention user studies to assess the impact and benefits of MDE for AI. For this, user studies are required to identify unused

potentials and further streamline the development towards a user-centered MDE for AI methodology.

6. Threats to Validity

The study's validity describes the extent to which the results are trustworthy and how biases arising from the subjective views of the researcher are avoided during the analysis. Validity must be considered at all stages of a study, and several approaches have been proposed in the literature. Following [35], the following threats to validity are considered:

- *Construct Validity:* Construct validity describes the validity of the concept or theory behind the study design such that the results are generalizable [65]. In this SLR, construct validity refers to the potentially subjective analysis of the studies and the different ways in which data extraction is conducted. Following the guidelines [35], each study analysis is conducted independently by at least two researchers. If the researchers cannot agree on a conclusion, a third researcher evaluates and discusses the literature until there is no disagreement. In addition, each selected literature was evaluated using the quality criteria suggested by [39]. A protocol based on [35] was defined for performing the extraction protocol, which was discussed by the performing researchers after each step.
- *Internal Validity:* Internal validity describes the causal relationships of the researcher's investigation of whether a factor influences an aspect under study. The particular danger is that a third factor has an unknown effect or side effect. To avoid this danger, the same behavior as for construct validity applies, that more than one researcher assesses the causal relationships. In addition, the tactic suggested by [35] was followed.
- *External Validity:* External validity exists when a finding in the selected literature is of interest to others outside the case under study. In this regard, the SLR uses a quality assessment based on [39], so included papers are published in peer-reviewed. Therefore, third-party investigators pre-assessed the selected studies, and the validity of the initial publication is the responsibility of the external authors.
- *Conclusion Validity:* The validity of the conclusion relates to concerns

about the reproducibility of the study. The concerns in this paper relate to the possible omission of studies. In this regard, the concerns are mitigated by the carefully applied search strategy using multiple digital libraries in conjunction with the snowballing system as per [35]. In addition, the researchers followed the detailed search protocol as defined in Section 3 and applied the quality ratings. However, some concerns might exist due to the interdisciplinary nature of the fields involved and the various definitions of modeling and AI. These were minimized, however, by the detailed background introduction in Section 2.

7. Conclusion

AI is emerging in several disciplines today and has recently attracted the interest of the MDE community, with several workshops being held on the subject. The development of AI requires several development phases, which potentially can be supported using MDE approaches. Currently, the support of AI by MDE is still at an early stage of development. Therefore, it is necessary to understand the existing approaches to support AI to streamline future research and build on existing knowledge.

We conducted an SLR to investigate the existing body of knowledge in MDE approaches to formalize and define AI applications. To this end, we followed a rigorous, SLR protocol, selected 15 approaches, and evaluated them for several dimensions of interest, from MDE and AI.

The result showed that the language engineering perspective of MDE for AI is already mature, and some approaches seem applicable in industrial case studies. The MDE approaches focus on the training phase of the AI approaches, while time-consuming tasks such as data pre-processing are not considered often. Additionally, the focus is not on improving communication, collaboration, or understanding of the business processes to be supported, which is reported in the literature as a core problem in AI development projects. Finally, the review showed that the approaches are case-specific and lack general applicability.

Future work in this research area is depicted in Section 5.6. It consists of, among other things, consolidating approaches to combine benefits, expanding approaches to be less case-specific, and adopting a closed-loop process that allows for model-based development that potentially leads to an authoritative

source of truth. We plan to develop a framework that allows us to mitigate some of the shortcomings and foster MDE with a focus on AI.

References

[1] Rama Akkiraju, Vibha Sinha, Anbang Xu, Jalal Mahmud, Pritam Gundecha, Zhe Liu, Xiaotong Liu, and John Schumacher. Characterizing Machine Learning Processes: A Maturity Framework. In Dirk Fahland, Chiara Ghidini, Jörg Becker, and Marlon Dumas, editors, *Business Process Management*, pages 17–31, Cham, 2020. Springer International Publishing.

[2] Issam Al-Azzoni. Model Driven Approach for Neural Networks. In *2020 International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*, pages 87–94, 2020.

[3] Ana Azevedo and Manuel Filipe Santos. KDD, SEMMA and CRISP-DM: A Parallel Overview. *IADIS European Conference Data Mining*, pages 182–185, 2008.

[4] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. The Goal Question Metric Approach. pages 1–10, 1994.

[5] B. Beihoff, C. Oster, S. Friedenthal, Christiaan Paredis, D. Kemp, H. Stoewer, D. Nichols, and J. Wade. A World in Motion – Systems Engineering Vision 2025. Technical report, INCOSE, San Diego, California, 2014.

[6] Massimo Bertolini, Davide Mezzogori, Mattia Neroni, and Francesco Zammori. Machine Learning for industrial applications: A comprehensive literature review. *Expert Systems with Applications*, 175:114820, August 2021.

[7] Anirban Bhattacharjee, Yogesh Barve, Shweta Khare, Shunxing Bao, Zhuangwei Kang, Aniruddha Gokhale, and Thomas Damiano. STRATUM: A BigData-as-a-Service for Lifecycle Management of IoT Analytics Applications. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1607–1612, 2019.

[8] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. Model-Driven Software Engineering in Practice: Second Edition. *Synthesis Lectures on Software Engineering*, 3(1):1–207, March 2017.

[9] Dominic Breuker. Towards Model-Driven Engineering for Big Data Analytics - An Exploratory Analysis of Domain-Specific Languages for Machine Learning. *47th Hawaii International Conference on System Sciences, HICSS 2014, Waikoloa, HI, USA, January 6-9, 2014*, pages 758–767, 2014.

[10] Matthias Brunnbauer, Gunther Piller, and Franz Rothlauf. Idea-AI: Developing a Method for the Systematic Identification of AI Use Cases. August 2021.

[11] Matthias Brunnbauer, Gunther Piller, and Franz Rothlauf. Top-Down or Explorative? A Case Study on the Identification of AI Use Cases. July 2022.

[12] Steven L. Brunton, J. Nathan Kutz, Krithika Manohar, Aleksandr Y. Aravkin, Kristi Morgansen, Jennifer Klemisch, Nicholas Goebel, James Buttrick, Jeffrey Poskin, Adriana W. Blom-Schieber, Thomas Hogan, and Darren McDonald. Data-Driven Aerospace Engineering: Reframing the Industry with Machine Learning. *AIAA Journal*, 59(8):2820–2847, 2021.

[13] Lola Burgueño, Jordi Cabot, Manuel Wimmer, and Steffen Zschaler. Guest editorial to the theme section on AI-enhanced model-driven engineering. *Software and Systems Modeling*, 21(3):963–965, June 2022.

[14] Loli Burgueño, Alexandru Burdusel, Sébastien Gérard, and Manuel Wimmer. MDE Intelligence 2019: 1st Workshop on Artificial Intelligence and Model-Driven Engineering. In *Proceedings of the 22nd International Conference on Model Driven Engineering Languages and Systems*, MODELS '19, pages 168–169. IEEE Press, 2021.

[15] Loli Burgueño, Federico Ciccozzi, Michalis Famelis, Gerti Kappel, Leen Lambers, Sebastien Mosser, Richard F. Paige, Alfonso Pierantonio, Arend Rensink, Rick Salay, Gabriele Taentzer, Antonio Vallecillo, and Manuel Wimmer. Contents for a Model-Based Software Engineering

Body of Knowledge. *Software and Systems Modeling*, 18(6):3193–3205, December 2019.

[16] Loli Burgueño, Marouane Kessentini, Manuel Wimmer, and Steffen Zschaler. MDE Intelligence 2021: 3rd Workshop on Artificial Intelligence and Model-Driven Engineering. *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 148–149, 2021.

[17] Krzysztof Czarnecki. Overview of generative software development. In *Unconventional Programming Paradigms: International Workshop UPP 2004, Le Mont Saint Michel, France, September 15-17, 2004, Revised Selected and Invited Papers*, pages 326–341. Springer, 2005.

[18] Christopher Davey, Sanford Friedenthal, Sky Matthews, David Nichols, Paul Nielsen, Christopher Oster, Taylor Riethle, Garry Roedler, Paul Schreinemakers, Emma Sparks, and Heinz Stoewer. Systems Engineering Vision 2035 – Engineering Solutions for a Better World. Technical report, INCOSE, San Diego, California, 2022.

[19] Alfonso de la Vega, Diego García-Saiz, Marta Zorrilla, and Pablo Sánchez. Lavoisier: A DSL for increasing the level of abstraction of data selection and formatting in data mining. *Journal of Computer Languages*, 60:100987, October 2020.

[20] Alican Dogan and Derya Birant. Machine learning and data mining in manufacturing. *Expert Systems with Applications*, 166:114060, March 2021.

[21] Frank Emmert-Streib and Matthias Dehmer. Defining Data Science by a Data-Driven Quantification of the Community. *Machine Learning and Knowledge Extraction*, 1(1):235–251, March 2019.

[22] P. Espadinha-Cruz, R. Godina, and E.M.G. Rodrigues. A review of data mining applications in semiconductor manufacturing. *Processes*, 9(2):1–38, 2021.

[23] Simon Fahle, Christopher Prinz, and Bernd Kuhlenkötter. Systematic review on machine learning (ML) methods for manufacturing processes – Identifying artificial intelligence (AI) methods for field application. *Procedia CIRP*, 93:413–418, 2020.

[24] M.M. Forootan, I. Larki, R. Zahedi, and A. Ahmadi. Machine Learning and Deep Learning in Energy Systems: A Review. *Sustainability (Switzerland)*, 14(8), 2022.

[25] Martin Fowler. *Domain Specific Languages*. Addison-Wesley Professional, 1st edition, 2010.

[26] Joan Giner-Miguelez, Abel Gómez, and Jordi Cabot. Describeml: A tool for describing machine learning datasets. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, MODELS '22, page 22–26, New York, NY, USA, 2022. Association for Computing Machinery.

[27] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.

[28] Thomas Hartmann, Assaad Moawad, Francois Fouquet, and Yves Le Traon. The next Evolution of MDE: A Seamless Integration of Machine Learning into Domain Modeling. In *Proceedings of the ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems*, MODELS '17, page 180. IEEE Press, 2017.

[29] Thomas Hartmann, Assaad Moawad, Cedric Schockaert, Francois Fouquet, and Yves Le Traon. Meta-Modelling Meta-Learning. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 300–305, 2019.

[30] Charles Hartsell, Nagabhushan Mahadevan, Shreyas Ramakrishna, Abhishek Dubey, Theodore Bapty, Taylor Johnson, Xenofon Koutsoukos, Janos Sztipanovits, and Gabor Karsai. Model-Based Design for CPS with Learning-Enabled Components. In *Proceedings of the Workshop on Design Automation for CPS and IoT*, DESTION '19, pages 1–9, New York, NY, USA, 2019. Association for Computing Machinery.

[31] Kaitlin Henderson and Alejandro Salado. Value and benefits of model-based systems engineering (MBSE): Evidence from the literature. *Systems Engineering*, 24(1):51–66, January 2021.

[32] T. Huldt and I. Stenius. State-of-practice survey of model-based systems engineering. *Systems Engineering*, 22(2):134–145, March 2019.

[33] Aníbal Iung, João Carbonell, Luciano Marchezan, Elder Rodrigues, Maicon Bernardino, Fabio Paulo Basso, and Bruno Medeiros. Systematic mapping study on domain-specific language development tools. *Empirical Software Engineering*, 25(5):4205–4249, September 2020.

[34] Nafiseh Kahani, Mojtaba Bagherzadeh, James R. Cordy, Juergen Dingel, and Daniel Varró. Survey and classification of model transformation tools. *Software and Systems Modeling*, 18(4):2361–2397, 2019.

[35] B. Kitchenham and S. Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical report, 2007.

[36] Barbara Kitchenham and Pearl Brereton. A systematic review of systematic review process research in software engineering. *Information and Software Technology*, 55(12):2049–2075, December 2013.

[37] Kaan Koseler, Kelsea McGraw, and Matthew Stephan. Realization of a Machine Learning Domain Specific Modeling Language: A Baseball Analytics Case Study. In Slimane Hammoudi, Luís Ferreira Pires, and Bran Selic, editors, *Proceedings of the 7th International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2019, Prague, Czech Republic, February 20-22, 2019*, pages 13–24. SciTePress, 2019.

[38] Evgeny Kusmenko, Svetlana Pavlitskaya, Bernhard Rumpe, and Sebastian Stuber. On the Engineering of AI-Powered Systems. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW)*, pages 126–133, San Diego, CA, USA, November 2019. IEEE.

[39] Hannah A Long, David P French, and Joanna M Brooks. Optimising the value of the critical appraisal skills programme (CASP) tool for quality appraisal in qualitative evidence synthesis. *Research Methods in Medicine & Health Sciences*, 1(1):31–42, September 2020.

[40] Levi Lúcio, Moussa Amrani, Juergen Dingel, Leen Lambers, Rick Salay, Gehan M.K. K Selim, Eugene Syriani, and Manuel Wimmer. Model transformation intents and their properties. *Software and Systems Modeling*, 15(3):647–684, 2016.

[41] Azad Madni and Shatad Purohit. Economic Analysis of Model-Based Systems Engineering. *Systems*, 7(1):12, February 2019.

[42] Sofia Meacham, Vaclav Pech, and Detlef Nauck. AdaptiveSystems: An Integrated Framework for Adaptive Systems Design and Development Using MPS JetBrains Domain-Specific Modeling Environment. *IEEE Access*, 9:127973–127984, 2021.

[43] Fran Melchor, Roberto Rodriguez-Echeverria, José M. Conejero, Álvaro E. Prieto, and Juan D. Gutiérrez. A Model-Driven Approach for Systematic Reproducibility and Replicability of Data Science Projects. In Xavier Franch, Geert Poels, Frederik Gailly, and Monique Snoeck, editors, *Advanced Information Systems Engineering*, Lecture Notes in Computer Science, pages 147–163, Cham, 2022. Springer International Publishing.

[44] Armin Moin, Moharram Challenger, Atta Badii, and Stephan Günnemann. A model-driven approach to machine learning and software modeling for the IoT: Generating full source code for smart Internet of Things (IoT) services and cyber-physical systems (CPS). *Software and Systems Modeling*, 21(3):987–1014, June 2022.

[45] Sergio Morales, Robert Clarisó, and Jordi Cabot. Towards a DSL for AI Engineering Process Modeling. *Product-Focused Software Process Improvement*, 13709:53–60, 2022.

[46] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1–18, August 2015.

[47] David Piorkowski, Soya Park, April Yi Wang, Dakuo Wang, Michael Muller, and Felix Portnoy. How AI Developers Overcome Communication Challenges in a Multidisciplinary Team: A Case Study. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW1):1–25, April 2021.

[48] Ivens Portugal, Paulo Alencar, and Donald Cowan. A Survey on Domain-Specific Languages for Machine Learning in Big Data. 2016.

[49] Foster Provost and Tom Fawcett. Data Science and its Relationship to Big Data and Data-Driven Decision Making. *Big Data*, 1(1):51–59, March 2013.

[50] S. Rädler and E. Rigger. A Survey on the Challenges Hindering the Application of Data Science, Digital Twins and Design Automation in Engineering Practice. *Proceedings of the Design Society*, 2:1699–1708, May 2022.

[51] Benoît Ries, Nicolas Guelfi, and Benjamin Jahic. An MDE Method for Improving Deep Learning Dataset Requirements Engineering using Alloy and UML. In Slimane Hammoudi, Luís Ferreira Pires, Edwin Seidewitz, and Richard Soley, editors, *Proceedings of the 9th International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2021, Online Streaming, February 8-10, 2021*, pages 41–52. SCITEPRESS, 2021.

[52] Eugen Rigger, Thomas Vosgien, Kristina Shea, and Tino Stankovic. A top-down method for the derivation of metrics for the assessment of design automation potential. *Journal of Engineering Design*, pages 1–31, October 2019.

[53] Alberto Rodrigues da Silva. Model-driven engineering: A survey supported by the unified conceptual model. *Computer Languages, Systems & Structures*, 43:139–155, 2015.

[54] Bernhard Rumpe, Katrin Hölldobler, and RWTH Aachen, editors. *MontiCore 5 Language Workbench*. Number Band 32 in Aachener Informatik-Berichte, Software-Engineering. Shaker Verlag, Aachen, edition 2017 edition, 2017.

[55] Jeff Saltz. CRISP-DM is Still the Most Popular Framework for Executing Data Science Projects, November 2020.

[56] L. Nelson Sanchez-Pinto, Yuan Luo, and Matthew M. Churpek. Big Data and Data Science in Critical Care. *Chest*, 154(5):1239–1248, November 2018.

[57] Iqbal H. Sarker. Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, 2(3):160, March 2021.

[58] Christoph Schröer, Felix Kruse, and Jorge Marx Gómez. A Systematic Literature Review on Applying CRISP-DM Process Model. *Procedia Computer Science*, 181:526–534, 2021.

[59] Pramila P. Shinde and Seema Shah. A Review of Machine Learning and Deep Learning Applications. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pages 1–6, Pune, India, August 2018. IEEE.

[60] Ida Someh, Barbara Wixom, and Angela Zutavern. Overcoming Organizational Obstacles to Artificial Intelligence Value Creation: Propositions for Research. pages 5809–5818, January 2020.

[61] Jakob Trauer, Sebastian Schweigert-Recksiek, Luis Onuma, Karsten Spreitzer, Markus Mörtl, and Markus Zimmermann. Data-Driven Engineering – Definitions and Insights from an Industrial Case Study for a New Approach in Technical Product Development. January 2020.

[62] Markus Voelter, Sebastian Benz, Christian Dietrich, Birgit Engelmann, Mats Helander, Lennart CL Kats, Eelco Visser, and GH Wachsmuth. Dsl engineering-designing, implementing and using domain-specific languages. 2013.

[63] Jens Westenberger, Kajetan Schuler, and Dennis Schlegel. Failure of AI projects: Understanding the critical factors. *Procedia Computer Science*, 196:69–76, January 2022.

[64] Rüdiger Wirth and Jochen Hipp. CRISP-DM: Towards a standard process model for data mining. *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, 2000.

[65] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering*. Springer Science & Business Media, June 2012.

[66] J. Xu, M. Kovatsch, D. Mattern, F. Mazza, M. Harasic, A. Paschke, and S. Lucia. A Review on AI for Smart Manufacturing: Deep Learning Challenges and Solutions. *Applied Sciences (Switzerland)*, 12(16), 2022.

[67] Julian Zucker and Myraeka d'Leeuwen. Arbiter: A Domain-Specific Language for Ethical Machine Learning. In Annette N. Markham, Julia Powles, Toby Walsh, and Anne L. Washington, editors, *AIES '20: AAAI/ACM Conference on AI, Ethics, and Society, New York, NY, USA, February 7-8, 2020*, pages 421–425. ACM, 2020.