



Use of Software Engineering methodological support for developing machine learning-based software: A Systematic Mapping Study


Mirna Muñoz

(Centro de Investigación en Matemáticas A.C., Unidad Zacatecas, Zacatecas, ZAC, 98068 México)
 <https://orcid.org/0000-0001-8537-2695>, mirna.munoz@cimat.mx)


Gabriel García-Mireles

(Universidad de Sonora. Blvd. Luis Encinas y Rosales s/n 83000 Hermosillo, México)
 <https://orcid.org/0000-0003-4633-7410>, mireles@mat.uson.mx)


Jezreel Mejía

(Centro de Investigación en Matemáticas A.C., Unidad Zacatecas, Zacatecas, ZAC, 98068 México)
 <https://orcid.org/0000-0003-0292-9318>, jmejia@cimat.mx)


Yadira Quiñonez

(Facultad de Informática Mazatlán, Universidad Autónoma de Sinaloa, Av. Universidad, Av. Leonismo Internacional y Tellería, 82000 Mazatlán, México)
 <https://orcid.org/0000-0002-7604-8532>, yadiraqui@uas.edu.mx)

Gloria Gasca-Hurtado

(Universidad de Medellín, Ingeniería en Sistemas, Facultad de Ingenierías, Carrera 87 No. 30-65, Medellín, Colombia)
 <https://orcid.org/0000-0003-0157-1959>, gpgasca@udemedellin.edu.co)

Adriana Peña

(Universidad de Guadalajara, CUCEI, Blvd. Marcelino García Barragán 1421, C.P. 44430, Guadalajara, Jal. Mexico)
 <https://orcid.org/0000-0001-6823-2367>, adriana.ppnegron@academicos.udg.mx)

Abstract: Advances in Artificial Intelligence (AI), particularly through Machine Learning (ML) and Deep Learning (DL), have enabled innovative solutions in industries including automotive, healthcare, and transportation. These technologies address complex problems by leveraging data at large scale for decision making. Nonetheless, their widespread adoption has also exposed critical technical challenges that can impact system performance in real world environments. Software Engineering (SE), a foundational discipline in Computer Science, provides methodological support to oversee the complete software development lifecycle, encompassing planning, design, implementation, testing, and maintenance. This renders SE relevant for developing reliable and maintainable systems based on AI. This paper conducts a Systematic Mapping Study (SMS) that analyzes 89 Primary Studies (PS) to investigate the current state of methodological support for SE in software development based on ML. Our findings reveal significant gaps in how SE practices are integrated into ML projects, with most research focusing narrowly on testing activities and overlooking other critical development stages. The study identifies core challenges and proposes recommendations organized according to the four dimensions of SE: process, product, project, and people. The results highlight the pressing need to refine and expand SE methodological sup-

port to address the distinct requirements of systems driven by AI, thereby enhancing reliability, maintainability, and ethical accountability in software development based on ML.

Keywords: Artificial Intelligence, Machine Learning based Software, Software Engineering, Systematic Mapping Study

Categories: H.3.1, H.3.2, H.3.3, H.3.7, H.5.1

DOI: 10.3897/jucs.147241

1 Introduction

AI is revolutionizing various sectors, including agriculture, education, healthcare, finance, transportation, defense, and manufacturing [Afzaal et al. 2024, Anastasiou et al. 2024, Huang et al. 2024, Kshetri 2024, Manta-Costa et al. 2024, Pasero 2024, Rettore et al. 2024]. The integration has resulted in substantial progress in work automation, data analysis, risk management, predictive maintenance, and decision-making processes [Gurjar et al. 2024, Rashid and Kausik 2024]. Organizations that utilize AI-driven solutions report enhancements in operational efficiency, service customization, and competitive advantage [Carleton et al. 2020]. Recent reports, like Accenture's Technology Vision 2025 [Accenture 2025], highlight the rapid acceleration of AI deployment and stress the increasing demand for digital systems that are dependable, transparent, and traceable.

Notwithstanding these advantages, the advancement of AI-based systems presents significant technical challenges that require robust engineering approaches. These challenges are particularly evident in systems powered by ML and DL technologies, which are extensively employed in fields such as autonomous vehicles, robotics, computational biology, and the Internet of Things (IoT) [Benton 2020, Nazir et al. 2024]. Principal concerns are data quality, model interpretability, overfitting, fairness, transferability, and legal and ethical considerations [IEEE 2024].

Although promising, ML-based systems can present challenges in testing, validation, and maintenance. They present unique issues such as limited model explainability, high computational demands for training, and unpredictable behavior under changing conditions [Deng 2018, Kumeno 2020, Ozkaya 2020]. These difficulties extend beyond the algorithmic level, affecting system architecture, development processes, quality assurance, and long-term maintenance [Dalpiaz and Niu 2020, Nazir et al. 2024].

SE plays a critical role in addressing these challenges by offering systematic methodologies for software specification, design, implementation, testing, and maintenance [Carbin 2019]. However, the degree to which SE practices are applied in the development of AI-based systems remains unclear. There is a pressing need to understand how SE can evolve to effectively support the development of ML-based systems, given their increasing complexity and impact [Busquim et al. 2024, Carbin 2019, Menzies 2020].

This paper presents an SMS to provide a comprehensive overview of SE methodological support in ML-based software development. Given ML's central role in contemporary AI systems, it is fundamental to identify the SE approaches currently employed and their limitations in order to improve system quality, reliability, and maintainability.

Our results indicate that, although substantial advancements have been made in SE for conventional software, their application in ML-based software development remains constrained. Most studies focus narrowly on testing, with other phases such as requirements engineering, system architecture, and maintenance receiving limited attention.

Through this study, we identify current structured methodological support used to develop ML-based software, structured around the four foundational dimensions of SE: Process, Product, Project, and People. Our analysis emphasizes the need for customized SE approaches that cater to the distinct characteristics of ML-based software.

The remainder of this paper is structured as follows: Section 2 reviews related work; Section 3 details the research methodology; Section 4 presents the main findings; Section 5 discusses key insights and implications; Section 6 addresses threats to validity; and Section 7 concludes the paper.

2 Background and related work

In recent years, an increasing number of research studies have examined the convergence of SE and AI, with a specific emphasis on the creation of ML-based software. This section examines 20 systematic literature reviews and surveys published between 2018 and 2022, which evaluate SE approaches used in AI/ML development and related problems.

2.1 General reviews on software engineering and artificial intelligence

A multitude of reviews offer comprehensive analyses of the interplay between SE and AI technologies [Colomo-Palacios 2020, Shehab et al. 2019, Syahputri et al. 2020, Martínez-Fernández et al. 2022, Ferreira et al. 2021, Kolltveit and Li 2022]. These studies generally focus on identifying overarching issues in implementing SE in ML-based systems, encompassing quality assurance, testing, and the integration of ML pipelines within software development processes. For example, [Kolltveit and Li 2022] analyzes the relevance of SE approaches in industrial settings and provides pragmatic suggestions for future endeavors. Likewise, [Martínez-Fernández et al. 2022] categorizes SE techniques using the SWEBOK framework [Bourque and Fairley 2014] and identifies 94 problems, noting that numerous challenges lie beyond existing SE standards, such as ISO/IEC 25010. Additional reviews focus on particular concerns, such as enhancing the interpretability of DL systems [Shehab et al. 2019] or tackling the challenges of testing and certifying ML applications [Colomo-Palacios 2020]. Furthermore, certain studies investigate the application of agile methodologies to ML projects, highlighting both their ubiquity and constraints, especially for data management and effectiveness evaluation [Ferreira et al. 2021, Syahputri et al. 2020].

2.2 Studies focused on technical challenges

A separate set of studies focuses on the technical problems related to ML-based software development [Kumeno 2020, Lwakatara et al. 2020, Saeed et al. 2021, Shivashankar and Martini 2022, Woehrle et al. 2019]. These efforts emphasize challenges about safety, security, verification, and explainability. For instance, [Saeed et al. 2021] examines difficulties in data management, model troubleshooting, deployment, and sustainability, whereas [Woehrle et al. 2019] focuses on testing challenges in safety-critical systems utilizing deep neural networks (DNNs). Other studies categorize obstacles based on development workflows or quality attributes, highlighting persistent issues such as adaptability, scalability, privacy, and usability [Lwakatara et al. 2020, Shivashankar and Martini 2022].

2.3 Domain-specific reviews and emerging topics

Certain studies focus on specialized areas within the SE and ML intersection. This includes evaluations of federated learning and its influence on software quality [Lo et al. 2021], equity in AI system development [Balayn et al. 2021], and the aspects of verification, explainability, and data quality in ML systems [Gezici and Tarhan 2022, Masuda et al. 2018]. Furthermore, several evaluations suggest design patterns and anti-patterns pertinent to ML software, emphasizing architectural and operational procedures [Washizaki et al. 2019, Washizaki et al. 2022]. Additional domain-specific research investigates requirements engineering for collaborative AI systems [Odong et al. 2022, Villamizar et al. 2021] and embedded systems, particularly in the context of IoT and TinyML [Lakshman and Eisty 2022].

2.4 Synthesis and research gaps

Several common themes arise across these investigations. Numerous studies focus on overarching difficulties, publication trends, or the categorization of SE operations within the framework of ML-based systems. Although some focus on particular concerns, such as quality assurance, explainability, or fairness, few offer comprehensive critiques of specific SE processes or methodological frameworks designed for ML or DL development.

A significant deficiency is the absence of detailed process models or lifecycle approaches tailored explicitly for ML-based software. Most current research either concentrates on particular development phases or remains at a theoretical level, lacking practical frameworks for practitioners.

This work comprehensively delineates the methodological support for SE utilized in ML-based software development. This paper presents a structured analysis of specific SE practices and methodological support employed throughout the ML software lifecycle, in contrast to previous evaluations that primarily address generic challenges or isolated topics. Additionally, it delineates significant research deficiencies and technical obstacles from a specific SE viewpoint, establishing a basis for forthcoming empirical investigations and practical progress in this field.

3 Research methodology

This study adopts the methodology and procedures described by [Petersen et al. 2008] to conduct a SMS. The primary objective of this SMS is to provide a structured and comprehensive overview of the existing literature regarding the application of SE practices and methodological support such as practices, methods, techniques or tools in the development of ML-based software.

3.1 Research questions and scope

The Research Questions (RQs) were formulated to explore both the extent and the characteristics of SE methodological support documented in the literature for ML-based software development. This includes identifying SE practices applied in real-world projects and cataloging the available methodological resources designed to guide such efforts. We distinguish between (i) Software Engineering practices, which are concrete,

repeatable activities empirically employed in ML projects (e.g., metamorphic testing, requirements modeling for ML, agile monitoring), and (ii) prescriptive methodological supports, which are formal artifacts that guide engineering work (models, methods, processes, standards, frameworks/guidelines/tools), regardless of current adoption.

In this study, “software” is broadly defined to include systems, applications, or services that incorporate AI technologies, such as ML or DL [Deng 2018, Quiñonez 2021].

The General Research Question (GRQ) guiding this SMS is: What Software Engineering methodological supports have been applied or proposed to develop ML-based software?

To address this overarching question, two specific RQs were defined:

- **RQ1.** *Which SE practices have been empirically used in developing ML-based software, and how are they distributed across lifecycle phases and contexts?*

Motivation: This question seeks to identify the empirical practices observed in PS on ML-based software development. A substantial amount of literature suggests methodological supports; however, the actual practices of teams remain ambiguous. Mapping reported concrete activities, including requirements modeling, testing techniques, agile ceremonies, and quality validation, allows for the identification of effectively covered lifecycle phases and areas of limited empirical adoption. This distinction offers insight into the current state of practice and identifies gaps in existing adoption.

- **RQ2.** *What prescriptive SE methodological supports have been proposed or tailored for ML-based software, and what validation do they report?*

Motivation: This question seeks to characterize the prescriptive landscape of methodological supports (models, methods, processes, standards, frameworks, guidelines, and tools) that were specifically developed or modified for ML-based SE. These supports, while not yet widely adopted, provide structured guidance aimed at addressing identified challenges in requirements, design, testing, deployment, and maintenance. Mapping these elements enables an assessment of their scope, type, and reported validation, such as conceptual proposals, proofs of concept, and empirical studies. This offers a systematic overview of the state of prescription and facilitates comparison with the actual practices identified in RQ1, thereby revealing the adoption gap between available resources and their application.

To guarantee methodological rigor, the study employed the PICOC framework [Kitchenham and Charters 2007] to guide the formulation of research questions and the creation of the search strategy. Table 1 presents the application of the PICOC framework in this study.

3.2 Identifying the relevant literature

3.2.1 Search strategy for primary studies

To identify relevant PS, we designed a search strategy based on established guidelines for systematic mapping studies. The SWEBOK Guide informed the selection of SE terms, Version 3.0 [Bourque and Fairley 2014], while the AI terms were derived from a group discussion that focused on ML and its associated categories. Table 2 lists the SE and AI terms considered in this study.

Population	Studies focusing on ML-based software, or those involving SE frameworks, approaches, processes, standards, models, methods, guidelines, lifecycle stages, and tools applied to the development of AI/ML components.
Intervention	The application, adaptation, or proposal of SE methods, practices, tools, processes, or standards for the development of AI/ML-based software.
Comparison	Not applicable. This review does not aim to compare interventions but to identify and categorize existing SE practices and methodological support in the AI/ML domain.
Outcome	Identification and synthesis of: (i) SE practices tailored or proposed for AI/ML-based development, (ii) technical challenges reported in the literature, (iii) relevant system properties and quality attributes.
Context	Studies conducted within the domains of SE and AI, with an emphasis on the development lifecycle of AI/ML systems.

Table 1: Application of the PICOC framework used in this study

SE Terms	Software engineering, software requirements, software design, software construction, software testing, software maintenance, software configuration management, software quality, software project, software process.
AI Terms	Artificial intelligence, machine learning, supervised learning, unsupervised learning, reinforcement learning, deep learning, evolutionary learning.

Table 2: Relevant SE and AI-related terms used in the search string.

We then combined these terms using the logical operator OR within their respective categories and the AND operator between categories to ensure comprehensive coverage of the intersection between SE and AI. The resulting search strings are summarized in Table 3.

SE Terms	“software engineering” OR “software requirements” OR “software design” OR “software construction” OR “software testing” OR “software maintenance” OR “software configuration management” OR “software quality” OR “software project” OR “software process”.
AI Terms	“artificial intelligence” OR “machine learning” OR “supervised learning” OR “unsupervised learning” OR “reinforcement learning” OR “deep learning” OR “evolutionary learning”.

Table 3: Search strings for SE and AI terms combined with the OR operator.

The search was executed in three widely recognized digital libraries: IEEE Xplore, ACM Digital Library, and Scopus. Table 4 presents the exact search strings adapted for each database and the number of retrieved records.

The inclusion and exclusion criteria were derived directly from the PICOC elements. Specifically, the “population,” defined as ML-based software and SE practices guided the selection of studies that addressed concrete SE processes or methods applied to AI development (see criteria IC1, IC7, and EC1). The “Outcome” informed the selection of studies reporting challenges, properties, or recommendations (IC7, EC6), while the “Context” justified the focus on studies within the SE and AI domains.

Database	String	Records
IEEE Xplore	("Abstract": "software engineering" OR "Abstract": "software requirements" OR "Abstract": "software design" OR "Abstract": "software construction" OR "Abstract": "software testing" OR "Abstract": "software maintenance" OR "Abstract": "software configuration management" OR "Abstract": "software quality" OR "Abstract": "software project" OR "Abstract": "software process") AND ("Abstract": "artificial intelligence" OR "Abstract": "machine learning" OR "Abstract": "supervised learning" OR "Abstract": "unsupervised learning" OR "Abstract": "reinforcement learning" OR "Abstract": "deep learning" OR "Abstract": "evolutionary learning")	1954
ACM DL	[[Abstract: "software engineering"] OR [Abstract: "software requirements"] OR [Abstract: "software design"] OR [Abstract: "software construction"] OR [Abstract: "software testing"] OR [Abstract: "software maintenance"] OR [Abstract: "software configuration management"] OR [Abstract: "software quality"] OR [Abstract: "software project"] OR [Abstract: "software process"]] AND [[Abstract: "artificial intelligence"] OR [Abstract: "machine learning"] OR [Abstract: "supervised learning"] OR [Abstract: "unsupervised learning"] OR [Abstract: "reinforcement learning"] OR [Abstract: "deep learning"] OR [Abstract: "evolutionary learning"]]	433
Scopus	(("software engineering" OR "software requirements" OR "software design" OR "software construction" OR "software testing" OR "software maintenance" OR "software configuration management" OR "software quality" OR "software project" OR "software process") AND ("artificial intelligence" OR "machine learning" OR "supervised learning" OR "unsupervised learning" OR "reinforcement learning" OR "deep learning" OR "evolutionary learning")) Refined by research area: (COMPUTER SCIENCE)	1954

Table 4: Search strings and number of records retrieved from each digital library. Boolean expressions were adapted to the syntax of each database.

3.2.2 Primary and secondary search process

The search procedure to identify relevant studies followed a systematic and structured approach, conducted using the search engines and filters provided by the selected databases. The process included the following steps:

1. Execute searches in the Title, Abstract, and Keywords fields of the databases.
2. Refine searches by limiting results to the Abstract field to enhance precision.
3. Restrict the search to the publication period from 2010 to 2023, in alignment with prior systematic reviews in this area [Martínez-Fernández et al. 2022].
4. Apply search strings and relevant filters in the IEEE Xplore, ACM Digital Library, and Scopus databases.

3.3 Paper selection criteria

3.3.1 Inclusion and exclusion criteria

To ensure the relevance and rigor of the selected studies, we defined a set of inclusion and exclusion criteria aligned with the PICOC framework. These criteria guided the

Inclusion Criteria	Exclusion Criteria
IC1. The paper addresses approaches, frameworks, standards, processes, activities, tasks, methods, models, guidelines, tools, lifecycle stages, and practices to develop ML-based software. (SEforAI).	EC1. The paper addresses the topic of Artificial Intelligence for Software Engineering (AIforSE).
IC2. The paper is a conceptual paper or an empirical paper.	EC2. The paper does not address any topic defining approaches, frameworks, standards, processes, activities, tasks, methods, models, guidelines, tools, lifecycle stages, and practices.
IC3. The paper is a secondary study (systematic literature review, mapping study, or literature survey).	EC3. The paper is written in a language distinct from English.
IC4. The paper is written in English.	EC4. The paper was published before 2010.
IC5. The paper is peer-reviewed (journal, conference, workshop).	EC5. The paper is an editorial, thesis, book, poster, and extended abstract, among other non-peer-reviewed documents.
IC6. The paper is published between 2010 and 2023.	EC6. The paper does not address a specific topic of ML.
IC7. The paper analyzed challenges, barriers, and issues related to using or developing SE approaches, frameworks, standards, processes, activities, tasks, methods, models, guidelines, tools, lifecycle stages, and practices to develop ML-based software.	EC7. The full paper is not available in the consulted databases.

Table 5: Inclusion and Exclusion Criteria

filtering process at both the title and abstract screening stages. The specific criteria used in this study are summarized in Table 5.

Papers focused on using AI or ML to improve SE practices (AIforSE) were excluded, as our focus is specifically on the application of SE practices in ML-based software development (SEforAI).

3.3.2 Filtering of the papers

The paper selection process consisted of multiple filtering stages, as follows:

- **Initial Screening:** A total of 505 papers were retrieved from the databases and imported for screening.
- **First Review:** Secondary studies (systematic literature reviews, mapping studies, and surveys) were identified and retained, resulting in 483 papers.
- **Second Review:** Duplicate records were removed, leaving 399 unique papers.

- **Third Review:** Inclusion and exclusion criteria were applied to titles, abstracts, and keywords. Papers clearly not meeting the criteria were excluded. If relevance was unclear, the full text was reviewed. After this step, 212 papers were retained.
- **Fourth Review:** During the data extraction phase, some papers were excluded because they failed to adequately address the research questions. The final selection consisted of 89 PS.

3.4 Quality checklist and procedures

3.4.1 Quality assessment of primary papers

To guarantee the rigor and dependability of our SMS, we implemented a systematic quality assessment procedure for all chosen primary research. This process aimed to assess the methodological rigor and reporting clarity of the included studies.

The evaluation was performed autonomously by the six co-authors of this study. Each manuscript was assessed by two evaluators utilizing a predetermined checklist. Disagreements were addressed in review meetings to achieve consensus. In the absence of a consensus, a third reviewer acted as an adjudicator.

Although no formal inter-rater reliability measure (e.g., Cohen's Kappa) was calculated, we maintained consistency through systematic discussions and recorded consensus judgments during the review process.

The quality checklist was adapted from established guidelines for empirical SE research [Hidellaarachchi et al. 2022], and included six evaluation items. The items evaluated critical elements, including the clarity of study aims, methodological rigor, data collection and analysis techniques, and the comprehensiveness of the presented findings.

Each study was evaluated using a five-point Likert scale, with ratings from 1 (poor) to 5 (excellent) for each criterion, as seen in Table 6. Studies having an average score below 3.0 across all criteria were excluded from the synthesis to maintain methodological rigor and reliability.

Item	Evaluation Criterion	Score (1 to 5)
1	Is the paper highly applicable to our SMS research focus?	
2	Is there a clear statement of the aim of the research?	
3	Is there a review of key past work?	
4	Is there a clear methodology for the research that is aligned with key research questions claimed for the research?	
5	Does the paper provide adequate information regarding the data collection and data analysis of the research?	
6	Are the research findings clearly stated and supported by the research questions?	

Table 6: Instrument for quality assessment of selected PS (adapted from [Hidellaarachchi et al. 2022]).

3.5 Data extraction strategy

To ensure traceability, reproducibility, and consistency throughout the review process, we adopted a structured and collaborative approach to data management. All metadata and extracted data from the selected PS were organized in a shared Google Spreadsheet, allowing collaborative editing, version control, and centralized access for all authors. The spreadsheet was structured with predefined columns corresponding to the twelve data extraction items listed in Table 7, such as reference, publication year, contribution type, and SWEBOK knowledge area.

No.	Field	Description
1	Reference	Complete citation of the study (authors, title, venue, year).
2	Publication Year	Year in which the primary study was published.
3	Source	Type and title of the publication venue (e.g., journal, conference, workshop).
4	Objective	Stated goal or research aim of the study.
5	Contribution Category	Nature of the study's contribution (e.g., challenge, taxonomy, model, guideline, process).
6	Specific Contributions	Concrete outcomes or proposed methods/models described in the study.
7	SWEBOK KA	Mapped SE knowledge areas (e.g., software requirements, testing, quality).
8	AI Technology	Type of AI used or discussed (e.g., ML, DL, fuzzy logic, or general AI).
9	Paper Type	Classification based on the type of publication (e.g., review, empirical, conceptual).
10	Research Method	Methodology used in the study (e.g., case study, survey, experiment).
11	Contextual Data	Domain, application area, and participant details if applicable.
12	Results	Summary of main findings, insights, or conclusions.

Table 7: Data extraction fields used for metadata management and synthesis.

Each entry was independently reviewed and validated by at least two authors to minimize bias and ensure consistency. Any discrepancies were resolved through regular consensus meetings. The spreadsheet also served as the central repository for recording quality assessment scores, classification tags, and summary notes for each study.

4 Findings

Data from the final set of 89 filtered PS were extracted for analysis and presentation. All chosen PS were released from 2011 to 2023. The studies show a particular preference for conference publications over journal articles during this period. Specifically, 46 papers (51.89%) were published in conference proceedings, while 29 (32.58%) appeared in journals. Other publication types had lower representation, comprising workshops (4 papers, 4.49%), symposium (6 papers, 6.74%), and book chapters (4 papers, 4.49%).

The following subsections present the main findings about the research questions.

4.1 Which SE practices have been empirically used in developing ML-based software, and how are they distributed across lifecycle phases and contexts? (RQ1)

Among the 89 PS, eight explicitly report SE practices for ML-based software development (9%). Table 8 summarizes these practices by software development phase. Testing-oriented practices predominate (five PS), focusing on differential testing, metamorphic testing, and property-based testing for DL components. Only one study addresses requirements modeling (RE4ML/RE4AL), while another discusses the application of agile monitoring artifacts, including daily stand-ups and sprint retrospectives, to solve challenges in ML projects. No studies reported SE practices for deployment or maintenance activities. These findings highlight a significant gap in post-deployment phases, which future research should aim to address.

In addressing RQ1, the current literature primarily focuses on testing practices, while practices related to requirements engineering, project monitoring, and ethics are still in their early stages of development. Practices for deployment, maintenance, and software evolution are virtually absent.

Software Development phase	Practice	Target
Requirements	– Requirements Modeling for ML and Actors-with-Learning (RE4ML) [Yu 2021]	Modeling ML systems
Testing	<ul style="list-style-type: none"> – Differential testing and Metamorphic testing [Ahuja et al. 2020] – A set of heuristics to help detect faulty gradients, Property-based testing, Metamorphic testing, Mutation testing, Coverage-Guided Fuzzing, and Proof-based testing (mentioned in [Braiek and Khomh 2020]) – Proof-based testing (mentioned in [Braiek and Khomh 2020]) – White-box testing [Trujillo et al. 2020] 	<ul style="list-style-type: none"> – Testing DL – Testing ML programs – Testing deep neural networks
Monitoring and control	<ul style="list-style-type: none"> – Daily stand-up meetings, sprint planning meetings [Singla et al. 2018] – Sprint retrospective [Singla et al. 2018] 	Analyze project issues for several ML projects
Quality	– Quality validation approach/method meetings [Tao et al. 2019]	– Validate the quality of ML systems
Other	– Fairway method [Chakraborty et al. 2020]	– Detection and mitigation of discrimination or ethical bias

Table 8: Practices summary.

As observed, most practices focus on testing ML-based software, as reported in [Ahuja et al. 2020, Braiek and Khomh 2020, Trujillo et al. 2020]. One study proposed a practice centered on modeling ML systems [Yu 2021]. At the same time, another addressed monitoring and controlling ML software issues by applying agile methods, such as stand-up meetings and retrospectives [Singla et al. 2018]. Additional work proposed an approach for improving the quality of ML systems [Tao et al. 2019]. Lastly, one study introduced practices for detecting and mitigating discrimination or ethical bias in ML systems [Chakraborty et al. 2020].

4.2 What prescriptive SE methodological supports have been proposed or tailored for ML-based software, and what validation do they report? (RQ2)

To address this question, we classified the contributions identified in the selected studies into four categories, based on the type of methodological support applied in the development of ML-based software:

1. **Models:** Conceptual descriptions that define relationships among elements to facilitate understanding or to support the development of SE artifacts.
2. **Methods:** Specific procedures or techniques designed to accomplish a particular task or objective within the software development process.
3. **Processes:** Structured sequences of activities, tasks, and steps used to develop, operate, maintain, and evolve software.
4. **Other Resources:** Additional contributions such as frameworks, approaches, and methodologies that provide practical guidance for supporting ML software development.

4.2.1 Models

Among the 89 PS analyzed, only ten propose models specifically designed to address challenges associated with the introduction of ML systems. Most of these models emphasize quality assurance, while others focus on code optimization or address specific aspects of ML software behavior.

Quality models constitute the majority of these proposals. Several studies suggest leveraging or extending the ISO 25010 standard to guide quality evaluation in ML-based systems [Gu et al. 2020, Kuwajima and Ishikawa 2019, Nakamichi et al. 2020]. These models highlight the necessity of adapting traditional quality frameworks to the distinct characteristics of ML components. Additional contributions in this category include practical guidelines for quality assurance derived from empirical experience in AI software testing [Gao et al. 2019], recommendations from quality assurance engineers for AI systems [Fujii et al. 2020], and a training data quality model that extends the SQuaRE data quality model to account for training data characteristics [Nakajima 2019]. Another study proposes a meta-model-based verification approach, Neutralist, which aims to improve the verification of DL programs [Nikanjam and Khomh 2021].

In the area of code optimization, three studies propose models focused on enhancing the efficiency of data-driven applications [Carbin 2019]. One study explores the automatic identification and traceability of links between data, code, and ML components using mining software repositories techniques [Njomou et al. 2021], while another introduces

the AI-SEAL taxonomy, which classifies AI usage in SE according to application context, automation level, and technology [Feldt et al. 2018].

Additionally, one study presents a behavior mutation model for benchmarking bias mitigation methods in ML systems [Hort et al. 2021]. This contribution highlights the growing interest in addressing the ethical dimensions of ML, although it is currently underrepresented in the literature.

In summary, although these models address important issues such as quality assurance, optimization, and bias mitigation, the findings suggest that research on modeling practices for ML-based software is still in its early stages of development. Most proposals focus on narrow, domain-specific concerns rather than providing comprehensive frameworks suitable for broad adoption across many ML development contexts. Furthermore, the limited number of studies proposing models highlights an important gap in the current literature, highlighting the need for further research to develop generalizable and integrative modeling approaches for ML-based software.

4.2.2 Methods

Seven PS in our SMS propose specific methods aimed at improving the quality of ML-based software. These methods focus on different phases of the software development lifecycle, including requirements engineering, risk management, testing, and quality evaluation.

In the area of requirements engineering, one study introduces a two-layer data requirements modeling method designed to specify the data needs of ML systems during the early requirements analysis phase [Shao and Wang 2022]. This method addresses a critical gap, as traditional requirements engineering approaches often overlook the importance of data characteristics in ML systems.

Regarding risk management, one study proposes a method for analyzing product, service, and system risks, with an emphasis on ML-related uncertainties [Uchihira 2022]. This contribution reflects the growing need for systematic risk analysis frameworks capable of addressing the stochastic characteristics of ML components.

Testing represents the most extensively covered area among the identified methods. Several studies present novel techniques for improving testing processes in ML systems. These include methods for generating effective datasets using combinatorial testing approaches [Gladisch et al. 2020], the development of test scenarios and oracles to assess model robustness in industrial environments [Chatterjee et al. 2022], and metamorphic testing methods tailored for neural network models [Nakajima 2019]. One particularly innovative approach involves using a test transformation bandit technique to optimize the selection of follow-up test cases in metamorphic testing [Spieker and Gotlieb 2021]. Collectively, these methods demonstrate the community's emphasis on enhancing the reliability and robustness of ML systems using improved testing methodologies.

Finally, one study addresses the challenge of deriving quality characteristics and measurement methods for ML systems [Nakamichi et al. 2020]. This method emphasizes that quality attributes must be defined in accordance with the specific goals and contexts of the systems under development, which aligns with the need for flexible and adaptable quality models in ML-based software projects.

The identified methods demonstrate significant progress in various aspects of ML-based software development. However, the research remains fragmented, with most methods focusing on specific lifecycle stages, particularly testing, while other phases, such as maintenance and deployment, are inadequately examined. This observation

stresses the need for more comprehensive and integrated methodological frameworks that can support the entire development lifecycle of ML-based systems.

4.2.3 Processes

A primary challenge in the development of AI-based software is the absence of established lifecycle models or standardized processes, as numerous projects continue to be produced without formal guidance. This study makes a significant contribution by identifying procedures and lifecycle models designed explicitly for ML-based software development. We identified 12 key publications that address this gap, highlighting the necessity to modify traditional SE processes, tasks, and workflows to meet the specific requirements of ML-based systems. Table 9 summarizes the recommended methodologies, which are classified according to validation levels as follows: Conceptual, PoC (Proof of Concept), Experimental, Empirical, and combined labels (e.g., Conceptual/PoC) indicate that the contribution exhibits characteristics of more than one validation type.

4.3 Other identified resources

This SMS also revealed some supplementary resources suggested to facilitate ML-based software development. In total, we found seven frameworks, five approaches, and one methodology, each addressing distinct aspects of the development process.

Several frameworks concentrate on testing activities. For example, the Quote Framework aims to enhance robustness and fairness during testing [Chen et al. 2023]. In contrast, RobOT and MTGAN focus on testing and retraining DL models, particularly in image classification [Liu et al. 2020, Wang et al. 2021]. Alternative frameworks address broader concerns, including data requirements engineering, process lifecycle management, and transition from experimental pilots to practical implementations [Castellanos et al. 2021, Dey 2022, Jafari et al. 2021]. These resources illustrate an increasing acknowledgment of the necessity for specialized tools and organized processes to manage the complexity of ML-based systems [Ekmeffjord et al. 2022].

Approaches identified in the studies tend to focus on design and monitoring activities [Kourouklidis et al. 2021]. Some utilize model-driven engineering (MDE) principles to specify intended ML behavior or to include analytics into software design, particularly in specialized fields like IoT or healthcare [Moin et al. 2022, Stirbu et al. 2022, Zhu et al. 2022]. Alternative methodologies focus on testing automation and lifecycle management through agile practices, highlighting the adaptability required for ML-centric projects [Das et al. 2021].

Furthermore, we identified a single methodology, Systems/Software Engineering Guided Machine Learning (SEGML), which proposes a five-step process for integrating SE and data science activities, covering feasibility analysis, requirements elicitation, model design, testing, and contractual specifications for ML components [Alharbi and Bhattacharyya 2021].

Collectively, these 13 resources offer a fragmented but growing body of knowledge aimed at systematizing the development of ML-based software. Nonetheless, the domain remains devoid of cohesive lifecycle models or comprehensive process frameworks. Most approaches focus on isolated stages, such as testing or data management, rather than comprehensively addressing the entire development lifecycle.

Three principal themes emerged from the proposals: the necessity for structured models that guide ML development activities, robust data management practices, and

Author	Proposal	Pointed out/remarks	Validation
[Amershi et al. 2019]	Process based on the ML system architecture and requirements that should be considered during the ML design.	ML workflows are highly non-linear and contain several feedback loops.	Empirical (validated with industrial case studies)
[Nakamichi et al. 2020]	Method focused on identifying requirement definition issues to derive the quality characteristics and measurement methods for ML systems.	The quality characteristics depend on the system's goals under development.	Conceptual (proposed framework, no empirical validation)
[Poth et al. 2020]	Product risk evaluation questionnaire named EvAla.	Validate the AI software quality and to evaluate products or services risks.	Proof of Concept (tested through case examples, limited validation)
[Spieker and Gottlieb 2021]	Process resulting from conducting an empirical study on actual ML bugs.	Examine patterns of bugs and how they evolve.	Empirical (validated via bug analysis in real projects)
[Gao et al. 2019]	Test process and a classification-based test modeling for AI classification function testing.	Evaluating AI software quality.	Proof of Concept/Experimental (tested in controlled test environments)
[Nascimento et al. 2019]	Four-stage process for ML systems development.	Lack of a typical development process for ML systems.	Conceptual (proposed only, no validation reported)
[Falcini et al. 2017]	A process development life cycle for automotive software.	V model of Automotive SPICE 3.0 on the W model.	PoC (applied to the automotive domain, illustrative example)
[Falcini and Lami 2017, Hesenius et al. 2019]	A process for developing Data-Driven Applications that integrate tasks for ML/AI solutions.	V Overall software development effort.	Conceptual/Example (described approach, limited practical validation)
[Angel and Namin 2022]	Stages identified for AI/ML-based application development life cycle.	It uses an agile methodology establishing an instantiable, iterative, and incremental process model.	Conceptual/PoC (framework described, some examples reported)
[Kohl et al. 2021]	A process model adjusted for developing NLP applications: Standardized Modeling Process for Natural Language Processing (STAMP 4 NLP) that provides a development process, including roles, tasks, artifacts, and best practices.	It uses an agile methodology establishing an instantiable, iterative, and incremental process model.	PoC (validated in NLP application case)
[Granlund et al. 2021]	A pipeline for Continuous Delivery for Machine Learning (CD4ML).	Using this pipeline, a cross-functional team produces ML applications based on code, data, and models in small and safe increments that can be reproduced and reliably released at any time, in short adaptation cycles.	Empirical/PoC (validated with prototype implementation and industrial examples)

Table 9: Primary studies proposed a resource to solve the lack of ML development processes.

rigorous testing and risk assessment mechanisms. These themes closely correspond with the unique challenges of ML-based software, where data dependency, model uncertainty, and system complexity often exceed the scope of traditional SE methodologies. The evidence from PS indicates that modifying and enhancing traditional SE practices remains

a critical priority for forthcoming research in this domain.

4.3.1 Technological context of the selected studies

Although the primary focus of this SMS is on SE practices and other methodological support in ML-based software development, we also examined the technological context of the selected studies. Specifically, we analyzed the AI tools and technologies reported within the PS to provide additional context for understanding the SE practices described.

Among the 89 studies, only seven explicitly reported the use of specific AI tools or frameworks. These tools can be classified into four main categories: libraries, algorithms, programming languages, and ML experiment management tools, as summarized in Table 10.

AI tools type	AI Tools
Library	PyTorch, Jupyter Notebooks, Keras, TensorFlow, CNTK, Theano, Torch, Caffe, Scikit-Learn, Pandas, NumPy and SciPy [Reimann and Kniesel-Wünsche 2020, Santhanam 2020, Zhang et al. 2022]
Algorithms	Classification, regression, clustering, dimensionality reduction, decision tree algorithms, convolutional Neural Network and Recurrent Neural Network architectures are the most used [Ferenc et al. 2020, Reimann and Kniesel-Wünsche 2020, Santhanam 2020].
Language	Python [Nakajima 2019, Njomou et al. 2021, Reimann and Kniesel-Wünsche 2020].
ML experiment management tools	Neptune.ai, MLflow, DVC and GitHub.

Table 10: AI technologies presented in PS.

Our analysis indicates that while specialized ML libraries such as TensorFlow, PyTorch, Keras, and Scikit-Learn are widely known, their reported use in SE-focused studies remains limited. Instead, most studies continue to rely on fundamental algorithms such as classification and clustering, which remain foundational across various AI applications.

Python stands out as the predominant programming language due to its extensive support for AI development, ease of use, and integration with a wide range of libraries and tools. However, despite its popularity, few studies have addressed challenges related to integrating these technologies into SE processes.

Additionally, we observed low adoption of ML experiment management tools such as MLflow, DVC, and Neptune.ai within the reviewed studies. These platforms, while offering valuable features for experiment tracking, reproducibility, and collaborative development, are not yet widely reported in the SE research addressing ML-based systems.

This analysis suggests that there is an opportunity for future studies to explore the intersection of SE practices with emerging MLOps tools and AI technologies, particularly in areas related to reproducibility, version control, and collaborative workflows.

5 Discussion

This SMS identifies several critical aspects that influence the evolution of SE practices in the context of ML-based software development. In particular, we identified several fundamental gaps in current research that present opportunities for future studies.

The analysis of validation levels indicates that the majority of methodological supports are confined to the Conceptual or Proof of Concept stage, with few demonstrations or illustrative examples available. A limited number achieved the Experimental level, and only a few presented Empirical evidence in practical settings. This gap highlights the limited adoption in practice, particularly in the areas of requirements engineering and maintenance.

One of the most evident gaps lies in the lack of robust models to guide the development of ML-based software products. Despite attempts to adapt existing standards, such as ISO 25010, to encompass quality aspects specific to ML systems, a clear need remains for standardized models that provide actionable guidance for quality assurance throughout the ML software lifecycle. This gap is particularly significant in small development teams, which often lack the resources and expertise to develop high-quality ML solutions.

Likewise, we noted a limited presence of well-defined methods that provide structured procedures for the development of ML software. Although numerous frameworks and tools are available for training models and deploying them in production, there is a lack of comprehensive methodological guidance covering the entire software lifecycle, from requirements to maintenance.

At the process level, we did not identify any generic processes that provide a cohesive set of interrelated activities for developing ML software products. Considering the complexity of ML systems, a generic process model must integrate key aspects, including risk management, short development cycles, continuous feedback, and thorough quality management, that address data, models, development artifacts, and runtime behavior.

Furthermore, it is fundamental to systematically map and adapt SE practices that can effectively support ML software development. Such practices must encompass all phases of development, including data management, model design and training, software integration, validation, and quality assurance.

Our analysis of challenges and recommendations revealed that the difficulties encountered in ML-based software projects encompass the traditional dimensions of SE, commonly referred to as the "4 Ps": Process, Product, Project, and People, along with an additional dimension, Ethics. Process-related challenges primarily involve the complexity of managing iterative, data-driven operations and maintaining quality in rapidly evolving systems. Product-related challenges center on the unique technical characteristics of ML solutions, including their reliance on data quality and the complexity of maintaining consistent system behavior. Project-related concerns highlight the management of development effort, resources, and risks in environments characterized by uncertainty and rapid technological change. People-related challenges point to the need for interdisciplinary teams with skills in both SE and AI, as well as the importance of training and communication practices to facilitate collaboration. Ultimately, while ethical considerations are often overlooked, they reveal the pressing need for greater attention to fairness, accountability, and transparency in ML-based systems.

These findings reveal the urgent necessity for an integrated research agenda that connects SE and ML by developing models, methods, processes, and practices, particularly designed for the context of ML-driven software development.

5.1 Technical challenges reported in the primary studies

Among the 89 PS examined, 20 specifically addressed technical challenges related to the development of ML-based software. These challenges were grouped into five categories: Process, Product, Project, People, and Ethics, with the majority of challenges concentrated in the Process and Product categories.

The process category encompasses the most extensive set of challenges. Many studies emphasize the difficulty of ensuring quality assurance throughout the development lifecycle, particularly in requirements engineering and testing phases [Arpteg et al. 2018, Belani et al. 2019, Marijan et al. 2019, Wan et al. 2021]. Common issues include the absence of standardized lifecycle models for ML-based systems and the lack of integrated frameworks to guide development [Balayn et al. 2021, Fischer et al. 2021, Haakman et al. 2021]. Research indicates challenges in implementing feature-based modeling [Apel et al. 2022] and establishing development guidelines [Fischer et al. 2020, Nguyen-Duc et al. 2020]. Moreover, human-related factors such as team decision-making, project scheduling, and version control pose considerable problems [Ahuja et al. 2020, Njomou et al. 2022, Wolf and Paine 2020].

In the product category, the challenges revolve around technical aspects of ML systems, such as handling high-dimensional data, ensuring prediction accuracy, and addressing fault localization [Ahuja et al. 2020, Apel et al. 2022, Balayn et al. 2021, Chen et al. 2020]. Additional concerns include the trustworthiness of model outputs, managing unexpected behavior, incompatibilities between ML libraries, and the limited portability of code across environments [Kim 2020, Lenarduzzi et al. 2021]. Furthermore, some studies discuss difficulties in integrating learning components with traditional system modules and challenges in using visual tools that align with strict system specifications [Bennaceur and Meinke 2018]. Deployment challenges in specialized domains, such as vision-based systems, were also noted [Ahuja et al. 2022, Hains et al. 2018].

The Project category focuses on the complexity of managing ML projects, which often require highly specialized solutions and computational resources [Bennaceur and Meinke 2018, Renggli et al. 2019, Wan et al. 2021]. Key challenges encompass establishing sufficient quality assurance processes, selecting effective testing methods, and integrating sensemaking practices to improve decision-making and system reliability [Hesenius et al. 2019, Marijan et al. 2019, Wolf and Paine 2020]. The maintenance of ML models, particularly in dynamic environments, was recognized as a significant concern [Sculley et al. 2015].

In the People category, the most frequently cited challenges relate to organizational and human factors [Arpteg et al. 2018, Khomh et al. 2018, Serban et al. 2020]. Studies consistently report a shortage of skills, training, and knowledge among software development teams, particularly in the context of AI technologies [Hill et al. 2016, Lenarduzzi et al. 2021, Nahar et al. 2022]. Further challenges include the absence of clear guidelines for integrating ML models into existing development processes and difficulties in team collaboration, particularly in areas such as communication, documentation, and engineering practices.

Although fewer studies addressed the ethics category, those that did highlighted significant gaps in ensuring the correctness and trustworthiness of ML and DL systems [Ahuja et al. 2020, Marijan et al. 2019]. Issues such as explainability, fairness, and accountability remain largely unresolved, posing risks for the implementation of these systems in practical environments [Fischer et al. 2020].

Our analysis shows that these challenges are not isolated to a specific phase of the software development lifecycle; instead, they recur throughout various stages, from

requirements engineering and design to testing, deployment, and maintenance. One of the most pressing concerns identified is ensuring the quality of ML models, which is particularly challenging due to the complexity and lack of interpretability inherent in these models. This points to an urgent need for more effective testing and validation approaches to ensure the reliability of ML-based systems prior to their deployment. Numerous research studies suggest methodologies, like metamorphic testing [Braiek and Khomh 2020] and differential testing [Ahuja et al. 2020], which enable error detection and behavioral verification under controlled conditions. Moreover, tools such as MLflow, which facilitate experiment tracking and model lifecycle management, are seen as essential for addressing traceability and supporting the continuous evolution of ML models over time.

5.2 Recommendations

Of the 89 PS examined, only 11 specifically offered recommendations for addressing challenges in the development of ML-based software. The recommendations were categorized into five categories: process, product, project, people, and ethics. The predominant recommendations are categorized under process, product, and project.

The Process category contains the most significant number of recommendations. Several studies emphasize the importance of incorporating feedback loops [Carleton et al. 2020, Hesenius et al. 2019, Ishikawa and Yoshioka 2019, Wan et al. 2021] into the development process to better manage the iterative nature of ML systems. Recommendations also include the adoption of mechanisms for quality control [Fujii et al. 2020] and for handling unexpected system behaviors [Ishikawa and Yoshioka 2019]. Additionally, some studies advocate for the integration of model-based and data-driven approaches [Harel et al. 2022, Wan et al. 2021] to enhance decision-making and process adaptability.

In the Product category, the recommendations primarily focus on the need to consider new types of requirements specific to ML solutions, as explicitly stated in [Carbin 2019, Sram 2011]. These include both functional and non-functional aspects, such as explainability and traceability. Furthermore, the development of tools for monitoring, tracing, and improving ML systems is highlighted as a critical need to ensure system robustness and transparency [Nascimento et al. 2019].

The Project category includes recommendations centered on applying established SE practices to ensure the quality of AI systems [Chakraborty et al. 2020, Dam et al. 2018, Gao et al. 2019, Nguyen-Duc et al. 2020, Menzies 2020]. Several studies suggest adapting traditional SE principles and integrating them into AI project workflows to enhance process reliability and product quality [Carleton et al. 2020]. There is also an emphasis on defining clear development guidelines tailored to the specific characteristics of AI projects.

The People category, although less frequently addressed, highlights the need for forming multidisciplinary teams. Studies recommend involving SE, data scientists, domain experts, and other specialists to ensure diverse perspectives and expertise [Hesenius et al. 2019]. Additionally, the importance of employing configuration management tools [Carleton et al. 2020] and providing adequate training for stakeholders to understand ML models and frameworks is emphasized [Sothilingam and Yu 2020].

No recommendations were found in the Ethics category among the examined papers. This absence highlights a substantial deficiency in the existing research landscape, as ethical considerations such as justice, accountability, and transparency are increasingly critical in AI development.

Based on the analysis of challenges and recommendations, it becomes evident that future research should prioritize developing resources and methodologies that support testing, quality evaluation, and assurance of ML-based systems. Such efforts are essential to ensure that ML solutions are not only practical but also reliable, maintainable, and aligned with both SE standards and ethical considerations.

6 Threats to validity

Despite the rigorous methodology applied in this study, several potential threats to validity must be acknowledged and carefully considered when interpreting the findings.

One potential source of bias is the selection process. Although we followed the established guidelines proposed by Kitchenham and Charters and conducted systematic searches in the most widely used digital libraries for SE and AI research, it is still possible that some relevant studies were inadvertently excluded. This could result from limitations in the search strings, indexing issues in the databases, or the inherent difficulty of retrieving studies from such an evolving and interdisciplinary field.

Another potential threat arises from reviewer bias during the study selection, data extraction, and quality assessment stages. To minimize this risk, we adopted a collaborative review process in which each study was independently evaluated by pairs of reviewers using standardized templates. However, subjective judgments may still have influenced the classification and interpretation of specific studies, particularly in cases where the scope or methodology was unclear.

Additionally, there is a risk of bias in data extraction and interpretation. Although we applied structured extraction forms and held regular calibration meetings to align our understanding, the analysis of complex studies may have been affected by the reviewers' academic backgrounds or disciplinary perspectives. Such influences could shape how specific findings were categorized or interpreted.

Limitations also exist regarding the quality assessment procedure. While we systematically applied a predefined quality checklist and resolved disagreements through consensus discussions, we did not compute formal statistical measures of inter-rater reliability, such as Cohen's Kappa. Consequently, although the assessment process was consistent and transparent, the level of agreement between reviewers was not quantified statistically.

Lastly, there are limitations related to the generalizability of this study. Our review focused exclusively on peer-reviewed literature published up to December 2023. As a result, industrial reports, grey literature, and emerging research that has not yet undergone formal peer review were excluded. This may limit the applicability of our findings to practical settings, especially given the rapid evolution of AI and ML technologies.

These considerations highlight the importance of interpreting our results with caution and stress the need for future empirical studies that can complement this mapping by exploring SE practices and challenges in real-world ML development contexts.

7 Conclusions

This SMS provides a structured overview of how SE practices have been applied to the development of ML-based software and identifies key technical challenges that arise across the software development lifecycle.

Our findings reveal that although all classical SE knowledge areas, such as requirements engineering, testing, quality management, and project management, are in principle adaptable to ML-based development, their practical application remains limited. Only 9% of the PS describe concrete SE practices used in real-world ML projects, with most efforts concentrated on testing. Practices associated with the early stages (such as requirements elicitation) and the later stages (including maintenance and evolution) are notably underrepresented, highlighting a considerable gap in the end-to-end integration of SE in ML-based systems.

The studies also reveal technical challenges that span multiple stages of the development lifecycle. These include the complexity of managing data dependencies, the stochastic nature of model behavior, difficulties in achieving reproducibility, and the lack of seamless integration between ML components and traditional SE workflows. Such challenges make evident the need for SE methodologies that are specifically tailored to the unique demands of ML-based systems and supported by appropriate tools and practices.

Regarding the tooling landscape, fewer than 10% of the studies report using specialized libraries or tools for experiment tracking, version control, or continuous delivery. Python is the dominant implementation language due to its ecosystem of AI libraries; however, the limited adoption of MLOps platforms and SE automation tools indicates a broader maturity gap in SE practices for ML development.

An additional layer of insight emerged when the findings were interpreted through the lens of the four foundational dimensions of SE: People, Product, Process, and Project, alongside ethical considerations. Three critical needs stand out. First, SE processes must be adapted to handle the evolving nature of data and model behavior. Second, iterative and feedback-driven development cycles are essential to align ML experimentation with structured SE practices. Third, a comprehensive approach to quality assurance is required, one that considers not only the software functionality but also ethical and non-functional attributes such as fairness, explainability, and trustworthiness.

This study highlights important implications for future research. There is a pressing need to build empirical evidence around the actual effects of SE practices on ML development outcomes. Future work should involve conducting in situ studies to assess how SE interventions influence the development of AI projects. Furthermore, the development of hybrid lifecycle models that bridge MLOps practices with traditional SE standards is necessary. Ethical principles should also be more tightly integrated into SE quality models, addressing current gaps in governance, bias mitigation, and transparency.

This SMS is subject to certain limitations. The review focused exclusively on peer-reviewed literature published up to December 2023, which may exclude recent industrial advancements or grey literature. As such, the results must be interpreted as indicative rather than exhaustive.

In future work, we plan to collaborate with both industry practitioners and academic researchers to document real-world SE practices in ML development. This effort will enable us to validate and refine the findings of this SMS, ultimately contributing to the creation of a grounded and actionable research agenda for developing reliable, maintainable, and ethically sound AI systems.

References

[Accenture 2025] Accenture (2025). Technology Vision 2025: AI: A Declaration of Autonomy. Accenture.

- [Afzaal et al. 2024] Afzaal, M., Shanshan, X., Yan, D., Younas, M. (2024). Mapping Artificial Intelligence Integration in Education: A Decade of Innovation and Impact (2013–2023)—A Bibliometric Analysis. *IEEE Access*, **12**, pp. 113275–113299.
- [Anastasiou et al. 2024] Fountas, S., Espejo-García, B., Kasimati, A., Gemtou, M., Panoutsopoulos, H., Anastasiou, E. (2024). Agriculture 5.0: Cutting-Edge Technologies, Trends, and Challenges. *IT Professional*, **26**(1), pp. 40–47.
- [Balayn et al. 2021] Balayn, A., Lofi, C., J., G. (2021). Managing bias and unfairness in data for decision support: a survey of machine learning and data engineering approaches to identify and mitigate bias and unfairness within data management and analytics systems. *The VLDB Journal*, **30**(5), pp. 739–768.
- [Benton 2020] Benton, W. C. (2020). Machine Learning Systems and Intelligent Applications. *IEEE Software*, **37**(4), pp. 43–49.
- [Bourque and Fairley 2014] Bourque, P., Fairley, R. E. (2014). Guide to the Software Engineering Body of Knowledge, Version 3.0. *IEEE Computer Society*.
- [Busquim et al. 2024] Busquim, G., Villamizar, H., Lima, M. J., Kalinowski, M. (2024). On the Interaction Between Software Engineers and Data Scientists When Building Machine Learning-Enabled Systems. In: Bludau, P., Ramler, R., Winkler, D., Bergsmann, J. (eds) *Software Quality as a Foundation for Security. SWQD 2024*. Lecture Notes in Business Information Processing, vol. 505. Springer, Cham.
- [Carbin 2019] Carbin, M. (2019). Overparameterization: A Connection Between Software 1.0 and Software 2.0. 3rd Summit on Advances in Programming Languages, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2019, pp. 1–13.
- [Carleton et al. 2020] Carleton, A. D., Harper, E., Menzies, T., Xie, T., Eldh, S., Lyu, M. R. (2020). The AI Effect: Working at the Intersection of AI and SE. *IEEE Software*, **37**(4), pp. 26–35.
- [Colomo-Palacios 2020] Colomo-Palacios, R. (2020). Cross Fertilization in Software Engineering. In *Systems, Software and Services Process Improvement: 27th European Conference, EuroSPI 2020*, Düsseldorf, Germany, September 9–11, 2020, Proceedings 27, Springer International Publishing, pp. 3–13.
- [Dalpiaz and Niu 2020] Dalpiaz, F., Niu, N. (2020). Requirements Engineering in the Days of Artificial Intelligence. *IEEE Software*, **37**(4), 7–10, July-Aug. 2020.
- [Deng 2018] Deng, L. (2018). Artificial Intelligence in the Rising Wave of Deep Learning: The Historical Path and Future Outlook [Perspectives]. *IEEE Signal Processing Magazine*, **35**(1), 177–180, Jan. 2018.
- [Ferreira et al. 2021] Ferreira, F., Silva, L. L., Valente, M. T. (2021). Software engineering meets deep learning. In *SAC '21: The 36th ACM/SIGAPP Symposium on Applied Computing*, Virtual Event Republic of Korea, New York, NY, USA: ACM.
- [Gezici and Tarhan 2022] Gezici, B., Tarhan, A. K. (2022). Systematic Literature Review on Software Quality for AI-Based Software. *Empirical Software Engineering*, **27**(3), p. 66.
- [Gurjar et al. 2024] Gurjar, K., Jangra, A., Baber, H., Islam, M., Sheikh, S. A. (2024). An Analytical Review on the Impact of Artificial Intelligence on the Business Industry: Applications, Trends, and Challenges. *IEEE Engineering Management Review*, **52**(2), pp. 84–102.
- [Hidellaarachchi et al. 2022] Hidellaarachchi, D., Grundy, J., Hoda, R., Madampe, K. (2022). The Effects of Human Aspects on the Requirements Engineering Process: A Systematic Literature Review. *IEEE Transactions on Software Engineering*, **48**(6), pp. 2105–2127, June 2022.
- [Huang et al. 2024] Huang, X., Huang, T., Gu, S., Zhao, S., Zhang, G. (2024). Responsible Federated Learning in Smart Transportation: Outlooks and Challenges. *IEEE Internet of Things Magazine*, **7**(5), pp. 22–28.
- [IEEE 2024] IEEE (2024). IEEE Guide for an Architectural Framework for Explainable Artificial Intelligence. *IEEE Std 2894-2024*, pp. 1–55, 30 Aug. 2024.

- [Kitchenham and Charters 2007] Kitchenham, B., Charters, S. (2007). Guidelines for Performing Systematic Literature Reviews in Software Engineering. *Technical Report EBSE 2007-001*, Keele University and Durham University, Joint Report, Durham, UK, pp. 1–51.
- [Kolltveit and Li 2022] Kolltveit, A. B., Li, J. (2022). Operationalizing Machine Learning Models - A Systematic Literature Review. 2022 IEEE/ACM 1st International Workshop on Software Engineering for Responsible Artificial Intelligence (SE4RAI), Pittsburgh, PA, USA, 2022, pp. 1–8.
- [Kshetri 2024] Kshetri, N. (2024). Generative Artificial Intelligence in the Financial Services Industry. *Computer*, **57**(6), pp. 102-108.
- [Kumeno 2020] Kumeno, F. (2020). Software Engineering Challenges for Machine Learning Applications: A Literature Review. *Intelligent Decision Technologies*, **13**(4), 463–476, February 2020.
- [Lakshman and Eisty 2022] Lakshman, S. B., Eisty, N. U. (2022). Software Engineering Approaches for TinyML Based IoT Embedded Vision: A Systematic Literature Review. In *Proceedings of the 4th International Workshop on Software Engineering Research and Practice for the IoT*, 2022, pp. 33–40.
- [Lo et al. 2021] Lo, S. K., Lu, Q., Wang, C., Paik, H. Y., Zhu, L. (2021). A Systematic Literature Review on Federated Machine Learning. *ACM Computing Surveys*, **54**(5), pp. 1–39.
- [Lwakatare et al. 2020] Lwakatare, L. E., Raj, A., Crnkovic, I., Bosch, J., Olsson, H. H. (2020). Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions. *Information and Software Technology*, **127**, p. 106368.
- [Manta-Costa et al. 2024] Manta-Costa, S., Araújo, O., Peres, R. S., Barata, J. (2024). Machine Learning Applications in Manufacturing—Challenges, Trends, and Future Directions. *IEEE Open Journal of the Industrial Electronics Society*, **5**, pp. 1085-1103.
- [Martínez-Fernández et al. 2022] Martínez-Fernández, S., et al. (2022). Software Engineering for AI-Based Systems: A Survey. *ACM Trans. Softw. Eng. Methodol.*, **31**(2), Article 37e (April 2022), 59 pages.
- [Masuda et al. 2018] Masuda, S., Ono, K., Yasue, T., Hosokawa, N. (2018). A Survey of Software Quality for Machine Learning Applications. In *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, Västerås, Sweden, pp. 279–284.
- [Menzies 2020] Menzies, T. (2020). The Five Laws of SE for AI. *IEEE Software*, **37**(1), pp. 81-85.
- [Nazir et al. 2024] Nazir, R., Bucaioni, A., Pelliccione, P. (2024). Architecting ML-enabled systems: Challenges, best practices, and design decisions. *J. Syst. Software*.
- [Odong et al. 2022] Odong, L. A., Perini, A., Susi, A. (2022). Requirements Engineering for Collaborative Artificial Intelligence Systems: A Literature Survey. In *International Conference on Research Challenges in Information Science*, Cham: Springer International Publishing, pp. 409–425.
- [Ozkaya 2020] Ozkaya, I. (2020). What Is Really Different in Engineering AI-Enabled Systems? *IEEE Software*, **37**(4), 3–6, July-Aug. 2020.
- [Pasero 2024] Pasero, E. (2024). Medicine 4.0: When New Technologies Work with Artificial Intelligence. *IEEE Instrumentation & Measurement Magazine*, **27**(1), pp. 63-66.
- [Petersen et al. 2008] Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M. (2008). Systematic Mapping Studies in Software Engineering. In *12th International Conference on Evaluation and Assessment in Software Engineering (EASE)*, BCS Learning & Development.
- [Petersen et al. 2015] Petersen, K., Vakkalanka, S., Kuzniarz, L. (2015). Guidelines for Conducting Systematic Mapping Studies in Software Engineering: An Update. *Information and Software Technology*, **64**, pp. 1–18.

- [Quiñonez 2021] Quiñonez, Y. (2021). An Overview of Applications of Artificial Intelligence Using Different Techniques, Algorithms, and Tools. In *Latin American Women and Research Contributions to the IT Field*, IGI Global, pp. 325–347.
- [Rashid and Kausik 2024] Rashid, A. B., Karim Kausik, M. A. (2024). AI Revolutionizing Industries Worldwide: A Comprehensive Overview of Its Diverse Applications. *Hybrid Advances*, 7, 100277.
- [Rettore et al. 2024] Rettore, P. H. L., Zibner, P., Alkhowaiter, M., Zou, C., Sevenich, P. (2024). Military Data Space: Challenges, Opportunities, and Use Cases. *IEEE Communications Magazine*, 62(1), pp. 70–76.
- [Saeed et al. 2021] Saeed, S., Abubakar, M. M., Karabatak, M. (2021). Software Engineering for Data Mining (ML-Enabled) Software Applications. In *2021 9th International Symposium on Digital Forensics and Security (ISDFS)*, Elazig, Turkey, pp. 1–9.
- [Shehab et al. 2019] Shehab, M., Abualigah, L., Jarrah, M. I., Alomari, O. A., Daoud, M. S. (2019). (AIAM2019) Artificial Intelligence in Software Engineering and Inverse: Review. *International Journal of Computer Integrated Manufacturing*, 33(10–11), 1129–1144.
- [Shivashankar and Martini 2022] Shivashankar, K., Martini, A. (2022). Maintainability Challenges in ML: A Systematic Literature Review. In *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Gran Canaria, Spain, pp. 60–67.
- [Syahputri et al. 2020] Syahputri, I. W., Ferdiana, R., Kusumawardani, S. S. (2020). Does System Based on Artificial Intelligence Need Software Engineering Method? Systematic Review. In *2020 Fifth International Conference on Informatics and Computing (ICIC)*, Gorontalo, Indonesia, pp. 1–6.
- [Villamizar et al. 2021] Villamizar, H., Escovedo, T., Kalinowski, M. (2021). Requirements Engineering for Machine Learning: A Systematic Mapping Study. In *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Palermo, Italy, pp. 29–36.
- [Washizaki et al. 2022] Washizaki, H., et al. (2022). Software-Engineering Design Patterns for Machine Learning Applications. *Computer*, 55(3), pp. 30–39, March 2022.
- [Washizaki et al. 2019] Washizaki, H., Uchida, H., Khomh, F., Guéhéneuc, Y.-G. (2019). Studying Software Engineering Patterns for Designing Machine Learning Systems. In *2019 10th International Workshop on Empirical Software Engineering in Practice (IWESEP)*, Tokyo, Japan, pp. 49–495.
- [Wieringa et al. 2006] Wieringa, R., Maiden, N., Mead, N., Rolland, C. (2006). Requirements Engineering Paper Classification and Evaluation Criteria: A Proposal and a Discussion. *Requirements Engineering*, 11, pp. 102–107.
- [Woehrle et al. 2019] Woehrle, M., Gladisch, C., Heinzemann, C. (2019). Open Questions in Testing of Learned Computer Vision Functions for Automated Driving. In *Computer Safety, Reliability, and Security: SAFECOMP 2019 Workshops, ASSURE, DECSoS, SASSUR, STRIVE, and WAISE*, Turku, Finland, Proceedings 38, Springer International Publishing, pp. 333–345.

A Annex Primary Studies

ID	Reference
(PS1)	[Yu 2021] Yu, E. (2021). Requirements Engineering for Actors-with-Learning: Encompassing the Two Kinds of Modeling for Full Cognitive Cycle RE.

(PS2)	[Ahuja et al. 2020] Ahuja, M., et al. (2020). Opening the software engineering toolbox for the assessment of trustworthy AI.
(PS3)	[Braiek and Khomh 2020] Braiek, H. B., Khomh, F. (2020). On testing machine learning programs.
(PS4)	[Trujillo et al. 2020] Trujillo, M., Linares-Vásquez, M., Escobar-Velásquez, C., Dusparic, I., Cardozo, N. (2020). Does Neuron Coverage Matter for Deep Reinforcement Learning?: A Preliminary Study.
(PS5)	[Singla et al. 2018] Singla, K., Bose, J., Naik, C. (2018). Analysis of Software Engineering for Agile Machine Learning Projects.
(PS6)	[Tao et al. 2019] Tao, C., Gao, J., Wang, T. (2019). Testing and Quality Validation for AI Software—Perspectives, Issues, and Practices.
(PS7)	[Chakraborty et al. 2020] Chakraborty, J., Majumder, S., Yu, Z., Menzies, T. (2020). Fairway: A Way to Build Fair ML Software.
(PS8)	[Reimann and Kniesel-Wünsche 2020] Reimann, L., Kniesel-Wünsche, G. (2020). Achieving guidance in applied machine learning through software engineering techniques.
(PS9)	[Santhanam 2020] Santhanam, P. (2020). Quality Management of Machine Learning Systems.
(PS10)	[Zhang et al. 2022] Zhang, H., Cruz, L., Deursen, A. v. (2022). Code Smells for Machine Learning Applications.
(PS11)	[Ferenc et al. 2020] Ferenc, R., Viskok, T., Aladics, T., Jász, J., Hegedűs, P. (2020). Deep-water framework: The Swiss army knife of humans working with machine learning models.
(PS12)	[Nakajima 2019] Nakajima, S. (2019). Dataset Diversity for Metamorphic Testing of Machine Learning Software.
(PS13)	[Njomou et al. 2021] Njomou, A. T., Bifona Africa, A. J., Adams, B., Fokaefs, M. (2021). MSR4ML: Reconstructing Artifact Traceability in Machine Learning Repositories.
(PS14)	[Idowu et al. 2022] Idowu, S., Osman, O., Strüber, D., Berger, T. (2022). On the Effectiveness of Machine Learning Experiment Management Tools.

(PS15)	[Nakamichi et al. 2020] Nakamichi, K., et al. (2020). Requirements-Driven Method to Determine Quality Characteristics and Measurements for Machine Learning Software and Its Evaluation.
(PS16)	[Gu et al. 2020] Gu, Z., Wang, J., Luo, S. (2020). Investigation on the quality assurance procedure and evaluation methodology of machine learning building energy model systems.
(PS17)	[Kuwanjima and Ishikawa 2019] Kuwanjima, H., Ishikawa, F. (2019). Adapting SQuaRE for Quality Assessment of Artificial Intelligence Systems.
(PS18)	[Gao et al. 2019] Gao, J., Tao, C., Jie, D., Lu, S. (2019). Invited Paper: What is AI Software Testing? and Why.
(PS19)	[Fujii et al. 2020] Fujii, C. G., et al. (2020). Guidelines for Quality Assurance of Machine Learning-Based Artificial Intelligence.
(PS20)	[Nikanjam and Khomh 2021] Nikanjam, A., Khomh, F. (2021). Design Smells in Deep Learning Programs: An Empirical Study.
(PS21)	[Carbin 2019] Carbin, M. (2019). Overparameterization: A Connection Between Software 1.0 and Software 2.0.
(PS22)	[Feldt et al. 2018] Feldt, R., de Oliveira Neto, F. G., Torkar, R. (2018). Ways of Applying Artificial Intelligence in Software Engineering.
(PS23)	[Hort et al. 2021] Hort, M., Zhang, J. M., Sarro, F., Harman, M. (2021). Fairea: A Model Behaviour Mutation Approach to Benchmarking Bias Mitigation Methods.
(PS24)	[Shao and Wang 2022] Shao, W., Wang, X. (2022). A Data Modeling Method for Machine Learning Systems.
(PS25)	[Uchihira 2022] Uchihira, N. (2022). Project FMEA for Recognizing Difficulties in Machine Learning Application System Development.
(PS26)	[Gladisch et al. 2020] Gladisch, C., Heinzemann, C., Herrmann, M., Woehrle, M. (2020). Leveraging Combinatorial Testing for Safety-Critical Computer Vision Datasets.
(PS27)	[Chatterjee et al. 2022] Chatterjee, A., Ahmed, B. S., Hallin, E., Engman, A. (2022). Testing of Machine Learning Models with Limited Samples: An Industrial Vacuum Pumping Application.

(PS28)	[Spieker and Gotlieb 2021] Spieker, H., Gotlieb, A. (2021). Summary of: Adaptive Metamorphic Testing with Contextual Bandits.
(PS29)	[Amershi et al. 2019] Amershi, S., et al. (2019). Software Engineering for Machine Learning: A Case Study.
(PS30)	[Poth et al. 2020] Poth, A., Meyer, B., Schlicht, P., Riel, A. (2020). Quality Assurance for Machine Learning – An Approach to Function and System Safeguarding.
(PS31)	[Nascimento et al. 2019] Nascimento, E. d. S., Ahmed, I., Oliveira, E., Palheta, M. P., Steinmacher, I., Conte, T. (2019). Understanding Development Process of Machine Learning Systems: Challenges and Solutions.
(PS32)	[Falcini et al. 2017] Falcini, F., Lami, G., Costanza, A. M. (2017). Deep Learning in Automotive Software.
(PS33)	[Falcini and Lami 2017] Falcini, F., Lami, G. (2017). Deep Learning in Automotive: Challenges and Opportunities.
(PS34)	[Hesenius et al. 2019] Hesenius, M., Schwenzfeier, N., Meyer, O., Koop, W., Gruhn, V. (2019). Towards a Software Engineering Process for Developing Data-Driven Applications.
(PS35)	[Angel and Namin 2022] Angel, S., Namin, A. S. (2022). Conceptual Mappings of Conventional Software and Machine Learning-Based Applications Development.
(PS36)	[Kohl et al. 2021] Kohl, P., Schmidts, O., Klöser, L., Werth, H., Kraft, B., Zündorf, A. (2021). STAMP 4 NLP – An Agile Framework for Rapid Quality-Driven NLP Applications Development.
(PS37)	[Granlund et al. 2021] Granlund, T., Stirbu, V., Mikkonen, T. (2021). Towards Regulatory-Compliant MLOps: Oravizio’s Journey from a Machine Learning Experiment to a Deployed Certified Medical Product.
(PS38)	[Chen et al. 2023] Chen, J., Wang, J., Ma, X., Sun, Y., Sun, J., Zhang, P., Cheng, P. (2023). QuoTe: Quality-oriented Testing for Deep Learning Systems.
(PS39)	[Wang et al. 2021] Wang, J., Chen, J., Sun, Y., Ma, X., Wang, D., Sun, J., Cheng, P. (2021). RobOT: Robustness-Oriented Testing for Deep Learning Systems.
(PS40)	[Liu et al. 2020] Liu, E., Huang, S., Zong, C., Zheng, C., Yao, Y., Zhu, J., Tang, S., Wang, Y. (2020). MTGAN: Extending Test Case Set for Deep Learning Image Classifier.

(PS41)	[Dey 2022] Dey, S. (2022). Evidence-Driven Data Requirements Engineering and Data Uncertainty Assessment of Machine Learning-Based Safety-Critical Systems.
(PS42)	[Jafari et al. 2021] Jafari, S. N., Besharati, M. R., Izadi, M., Hourali, M. (2021). SELM: Software Engineering of Machine Learning Models.
(PS43)	[Castellanos et al. 2021] Castellanos, S., Varela, C. A., Correal, D. (2021). ACCORDANT: A Domain-Specific Model and DevOps Approach for Big Data Analytics Architectures.
(PS44)	[Ekmefjord et al. 2022] Ekmefjord, S. M., et al. (2022). Scalable Federated Machine Learning with FEDn.
(PS45)	[Kourouklidis et al. 2021] Kourouklidis, S. P., Kolovos, D., Noppen, J., Matragkas, N. (2021). A Model-Driven Engineering Approach for Monitoring Machine Learning Models.
(PS46)	[Moin et al. 2022] Moin, S., Challenger, A., Badii, A., Günemann, S. (2022). 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).
(PS47)	[Stirbu et al. 2022] Stirbu, S., Granlund, T., Mikkonen, T. (2022). Continuous Design Control for Machine Learning in Certified Medical Systems.
(PS48)	[Zhu et al. 2022] Zhu, S., Long, T., Wang, W., Memon, A. (2022). Improving ML-Based Information Retrieval Software with User-Driven Functional Testing and Defect Class Analysis.
(PS49)	[Das et al. 2021] Das, S., et al. (2021). Agile Systems Engineering in Building Complex AI Systems.
(PS50)	[Alharbi and Bhattacharyya 2021] Alharbi, S. J., Bhattacharyya, S. (2021). Machine Learning with System/Software Engineering in Selection and Integration of Intelligent Algorithms.
(PS51)	[Arpteg et al. 2018] Arpteg, A., Brinne, B., Crnkovic-Friis, L., Bosch, J. (2018). Software Engineering Challenges of Deep Learning.
(PS52)	[Belani et al. 2019] Belani, S. H., Vukovic, M., Car, Ž. (2019). Requirements Engineering Challenges in Building AI-Based Complex Systems.
(PS53)	[Wan et al. 2021] Wan, Z., Xia, X., Lo, D., Murphy, G. C. (2021). How does Machine Learning Change Software Development Practices?

(PS54)	[Marijan et al. 2019] Marijan, D., Shang, W., Shukla, R. (2019). Implications of Resurgence in Artificial Intelligence for Research Collaborations in Software Engineering.
(PS55)	[Marijan and Gotlieb 2020] Marijan, D., Gotlieb, A. (2020). Software Testing for Machine Learning.
(PS56)	[Khomh et al. 2018] Khomh, F., Adams, B., Cheng, J., Fokaefs, M., Antoniol, G. (2018). Software Engineering for Machine-Learning Applications: The Road Ahead.
(PS57)	[Lenarduzzi et al. 2021] Lenarduzzi, V., Lomio, F., Moreschini, S., Taibi, D., Tamburri, D. A. (2021). Software Quality for AI: Where We Are Now?
(PS58)	[Xie et al. 2020] Xie, X., Zhang, Z., Chen, T. Y., Liu, Y., Poon, P. -L., Xu, B. (2020). METTLE: A METamorphic Testing Approach to Assessing and Validating Unsupervised Machine Learning Systems.
(PS59)	[Haakman et al. 2021] Haakman, M., Cruz, L., Huijgens, H., et al. (2021). AI lifecycle models need to be revised.
(PS60)	[Fischer et al. 2021] Fischer, L., Ehrlinger, L., Geist, V., Ramler, R., Sobieszky, F., Zellinger, W., Brunner, D., Kumar, M., Moser, B. (2021). AI System Engineering—Key Challenges and Lessons Learned.
(PS61)	[Balayn et al. 2021] Balayn, A., Lofi, C., J., G. (2021). Managing bias and unfairness in data for decision support: a survey of machine learning and data engineering approaches to identify and mitigate bias and unfairness within data management and analytics systems.
(PS62)	[Lwakatare et al. 2019] Lwakatare, L. E., Raj, A., Bosch, J., Olsson, H. H., Crnkovic, I. (2019). A Taxonomy of Software Engineering Challenges for Machine Learning Systems: An Empirical Investigation.
(PS63)	[Apel et al. 2022] Apel, S., Kästner, C., Kang, E. (2022). Feature Interactions on Steroids: On the Composition of ML Models.
(PS64)	[Nguyen-Duc et al. 2020] Nguyen-Duc, A., Sundbø, I., Nascimento, E., Conte, T., Ahmed, I., Abrahamsson, P. (2020). A Multiple Case Study of Artificial Intelligent System Development in Industry.
(PS65)	[Fischer et al. 2020] Fischer, L., et al. (2020). Applying AI in Practice: Key Challenges and Lessons Learned.

(PS66)	[Warnett and Zdun 2022] Warnett, S. J., Zdun, U. (2022). Architectural Design Decisions for Machine Learning Deployment.
(PS67)	[Serban and Visser 2022] Serban, A., Visser, J. (2022). Adapting Software Architectures to Machine Learning Challenges.
(PS68)	[Myllyaho et al. 2022] Myllyaho, L., Raatikainen, M., Männistö, T., Nurminen, J. K., Mikkonen, T. (2022). On misbehaviour and fault tolerance in machine learning systems.
(PS69)	[Wolf and Paine 2020] Wolf, C. T., Paine, D. (2020). Sensemaking Practices in the Everyday Work of AI/ML Software Engineering.
(PS70)	[Gao et al. 2022] Gao, K., Wang, Z., Mockus, A., Zhou, M. (2022). On the variability of software engineering needs for deep learning: Stages, trends, and application types.
(PS71)	[Njomou et al. 2022] Njomou, A. T., Fokaefs, M., Silatchom Kamga, D. F., Adams, B. (2022). On the Challenges of Migrating to Machine Learning Life Cycle Management Platforms.
(PS72)	[Chen et al. 2020] Chen, Z., Cao, Y., Liu, Y., Wang, H., Xie, T., Liu, X. (2020). A comprehensive study on challenges in deploying deep learning based software.
(PS73)	[Kim 2020] Kim, M. (2020). Software Engineering for Data Analytics.
(PS74)	[Bennaceur and Meinke 2018] Bennaceur, A., Meinke, K. (2018). Machine Learning for Software Analysis: Models, Methods, and Applications.
(PS75)	[Hains et al. 2018] Hains, G., Jakobsson, A., Khmelevsky, Y. (2018). Towards formal methods and software engineering for deep learning: Security, safety and productivity for DL systems development.
(PS76)	[Ahuja et al. 2022] Ahuja, M., Gotlieb, A., Spieker, H. (2022). Testing Deep Learning Models: A First Comparative Study of Multiple Testing Techniques.
(PS77)	[Renggli et al. 2019] Renggli, C., Hubis, F. A., Karlaš, B., Schawinski, K., Wuz, W., Zhang, C. (2019). Ease.ml/ci and Ease.ml/meter in action: Towards data management for statistical generalization.
(PS78)	[Sculley et al. 2015] Sculley, D., et al. (2015). Hidden technical debt in machine learning systems.
(PS79)	[Sothilingam and Yu 2020] Sothilingam, R., Yu, S. K. E. (2020). Modeling Agents, Roles, and Positions in Machine Learning Project Organizations.

(PS80)	[Serban et al. 2020] Serban, A., van der Blom, K., Hoos, H., Visser, J. (2020). Adoption and Effects of Software Engineering Best Practices in Machine Learning.
(PS81)	[Hill et al. 2016] Hill, C., Bellamy, R., Erickson, T., Burnett, M. (2016). Trials and tribulations of developers of intelligent systems: A field study.
(PS82)	[Nahar et al. 2022] Nahar, N., Zhou, S., Lewis, G., Kästner, C. (2022). Collaboration challenges in building ml-enabled systems: Communication, documentation, engineering, and process.
(PS83)	[Carleton et al. 2020] Carleton, A. D., Harper, E., Menzies, T., Xie, T., Eldh, S., Lyu, M. R. (2020). The AI Effect: Working at the Intersection of AI and SE.
(PS84)	[Ishikawa and Yoshioka 2019] Ishikawa, F., Yoshioka, N. (2019). How Do Engineers Perceive Difficulties in Engineering of Machine-Learning Systems? - Questionnaire Survey.
(PS85)	[Harel et al. 2022] Harel, D., Marron, A., Sifakis, J. (2022). Creating a Foundation for Next-Generation Autonomous Systems.
(PS86)	[Sram 2011] Sram, N. (2011). Practical application of fuzzy logic: Experiences and current state in software engineering.
(PS87)	[Menzies 2020] Menzies, T. (2020). The Five Laws of SE for AI.
(PS88)	[Chakraborty et al. 2020] Chakraborty, A., Bagavathi, R., Tomer, U. (2020). A Comprehensive Decomposition towards the Facets of Quality in IoT.
(PS89)	[Dam et al. 2018] Dam, H. K., Tran, T., Ghose, A. (2018). Explainable Software Analytics.