The Evolution of Technical Debt from DevOps to Generative AI: A Multivocal Literature Review

Sergio Moreschini[a,b], Elvira Arvanitou[c], Elisavet-Persefoni Kanidou[d], Nikolaos Nikolaidis[d], Ruoyu Su[b], Apostolos Ampatzoglou[d], Alexander Chatzigeorgiou[d], Valentina Lenarduzzi[b]

[a] *Tampere University, Tampere (Finland)*
[b] *University of Oulu, Oulu (Finland)*
[c] *International Hellenic University, Thessaloniki (Greece)*
[d] *University of Macedonia, Thessaloniki (Greece)*

Abstract

Background: The rapid integration of Artificial Intelligence (AI)—including Machine Learning (ML) and Generative AI—into software systems is reshaping the software development lifecycle. As AI-driven systems become more dynamic and complex, traditional approaches to Technical Debt (TD) management face increasing limitations. Simultaneously, AI-assisted development introduces new forms of TD, particularly in relation to maintainability, explainability, and data governance.

Objective: This study aims to explore how Technical Debt Management (TDM) must adapt in the context of AI-enhanced software development. It investigates (1) the evolution of TD in AI-driven systems, and (2) the implications of using AI technologies within the software engineering process.

Method: We conducted a multivocal literature review ~~(MLR)~~, combining insights from both peer-reviewed research and industry sources. Following established guidelines, we systematically analyzed 61 primary sources, categorized TD types and management activities, and identified key challenges and practices emerging in the AI era.

Results: Our findings reveal that data-related, infrastructure, and pipeline-related TD are particularly prevalent in ML systems. Machine Learning Operations (MLOps) practices are increasingly recognized as essential for managing such debt, especially in relation to dynamic data dependencies and model retraining. In parallel, AI-generated artifacts and automated pipelines introduce new governance and maintainability challenges.

Conclusion: Technical Debt in AI systems demands continuous, automated, and cross-functional management strategies. As software evolves in response to data and usage, new operational paradigms—grounded in practices like MLOps and Small Language Model Operations (SLMOps)—will be vital to ensure long-term software sustainability. This study provides a foundational map for researchers and practitioners navigating the intersection of AI and TD management.

Keywords: Technical Debt, Multivocal Literature Review

1. Introduction

With the rise of Artificial Intelligence (AI), the software engineering (SE) community faces new challenges but is also alerted to grab "new" and "very promising" opportunities [1]. The intersection of AI and SE in both research and practice has changed drastically in recent years, with the following broad disciplines being the most prominent ones:

- An entirely new domain in SE aims at the *development of AI-driven applications* tailoring existing software engineering practices and tools (SE4AI) [2]. According to ABI research[1], the Artificial Intelligence (AI) software market size was valued at 98 billion dollars in 2024. With a growing Compound Annual Growth Rate (CAGR)[2] of 30%, the AI software market is expected to reach a size of more than 391 billion dollars in 2030. Given the differences in the development lifecycle of AI software to traditional software, as well as their different quality needs, there is a need for specialized SE approaches for developing software that is driven by or uses AI.
- *Software engineers leverage AI to enhance or redesign software development approaches* (AI4SE) [2]. As more and more AI solutions (e.g., Co-Pilot, ChatGPT, etc.) are becoming available, software engineers use them to increase their productivity in various parts of the software engineering lifecycle (e.g., write code and tests, generate documentation, improve requirements specifications, identify hotspots, apply refactorings, etc.).
- *SE researchers use AI to improve their methods and produce knowledge* (AI4SE-Research) [3]. As Machine Learning (ML), Deep Learning (DL), Generative AI (GenAI), and more classical AI models are becoming available and understandable to SE researchers, they tend to increasingly use them to extend and validate their research contribu-

[1] https://www.abiresearch.com/news-resources/chart-data/report-artificial-intelligence-market-size-global

[2] `https://www.investopedia.com/terms/c/cagr.asp`

tions (e.g., using ML for identifying energy hungry code, or improve defect and vulnerability prediction).

A large portion of the cost of software systems and therefore their success, lies on the maintenance phase of the SDLC (accounting up to 50% of the total software costs). To reduce maintenance costs, the software development team needs to pay attention to quality even from the early stages of development. More specifically, when targeting the improvement of maintainability, an established notion for collectively handling software maintenance is Technical Debt Management (TDM). In this work, acknowledging the importance of quality for software systems regardless of their development strategy, we focus on TDM and tailor the notion of TDM to match the intersection of TDM and AI. In particular, we attempt through a Multivocal Literature Review (MLR) to provide a panorama of the intersection of the two domains. Given that AI4SE-Research has already been covered by other secondary studies (e.g., [4], [5]), in this work, we focus on SE4AI and AI4SE. Studying both directions of the relation between AI and SE is a common place in the call for papers of various conferences (e.g., Maltesque [3], FSE[4], ESEM[5], etc.), strengthening the intuition that these two aspects are interconnected and cannot be studied in isolation. Given this view, we have set the following main research questions that will drive the study design and reporting:

- **[RQ$_1$]** How does TDM need to be tailored to match the needs of developing AI-driven software?
- **[RQ$_2$]** How does the use of AI during software development affect TDM?

The answer to RQ$_1$ is expected to shed light on how TDM approaches and tools need to be adapted to match the development lifecycle of AI-driven software. For example, we expect to explore: (a) the special challenges of TDM in SE4AI, by exploring the research goals of papers; (b) the TDM activities that receive the most attention in SE4AI; and (c) any possible new types of Technical Debt (TD) that have emerged in SE4AI. The answer to RQ$_2$ will explore TDM-related benefits and consequences of using AI to develop software. For instance, we will explore the opportunities that open up by resorting to AI4SE, but also the negative impacts that AI4SE might bring in terms of code quantity, quality, maintainability, and therefore TD. The aforementioned contributions, along with the context and the problem statement, are visualized in Figure 1.

The rest of the paper is organized as follows. In Section 2, we present secondary studies in the field of TDM to outline the related work on the more generic domain of Technical Debt research, as well as secondary studies on the intersection of SE and AI. In Section 3, we present the study design, where we outline the MLR protocol. We present the results of our work in Section 4, and we discuss them (interpretations, comparison with the literature, and implications for research and practice) in Section 5. In Section 6, we report on the threats to validity and conclude the paper in Section 7.

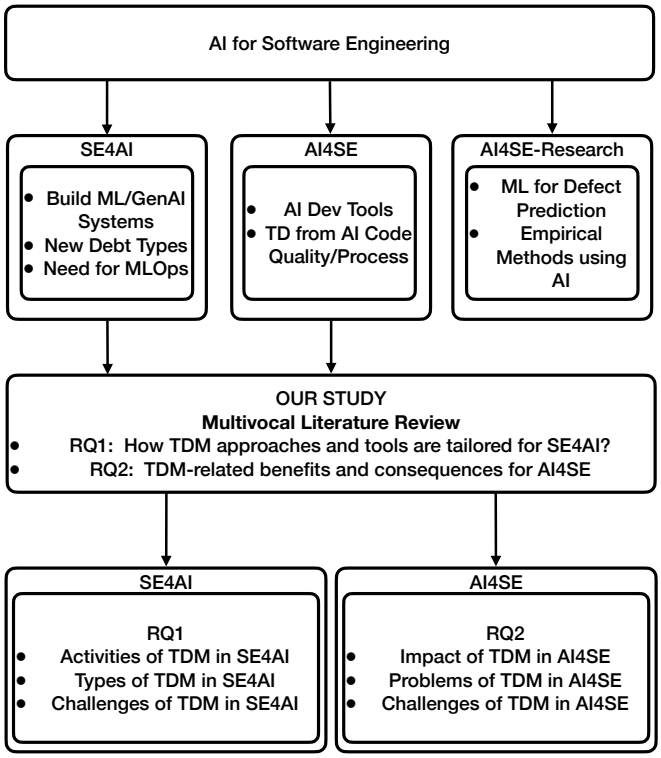

Figure 1: Context, Problem Statement, and Contribution of the Work

2. Related Work

In the last ten years, TD has been increasingly investigated by attracting the attention of both academia and industry. Several literature reviews and mapping studies in terms of TD and TDM have been published. In this section, we present secondary studies that are focused on TD. To identify such studies we have exploited recent tertiary studies on TDM[6][7][8], but have also conducted searches in the most known digital libraries–namely: ACM, IEEE, Springer, and ScienceDirect. Several secondary studies have explored TD in traditional software development, with a focus on architecture, prioritization, measurement, and tool support (see Table 1). However, most of these works do not address the impact of AI-driven development or modern practices like MLOps. In contrast, our study provides a multivocal and up-to-date perspective that specifically investigates how TD evolves in the context of AI-enhanced systems, making it among the first to bridge both SE4AI (RQ$_1$) and AI4SE (RQ$_2$) viewpoints.

Beskel et al. [9] conducted a Systematic Literature Review (SLR) to synthesize existing research on Architectural Technical Debt (ATD) and developed a descriptive model to classify and analyze ATD characteristics. In particular, the study focused on understanding and managing ATD in large-scale

[3] https://maltesque2022.github.io/

[4] https://conf.researchr.org/track/fse-2025/fse-2025-research-papers

[5] https://conf.researchr.org/track/esem-2025/esem-2025-vision-and-emerging-results-track-

Table 1: Overview of Secondary Studies on Technical Debt

Study	Focus Area	TD Type(s)	Methodology	Time Span	RQ
Beskel et al. [9]	Architectural Technical Debt	ATD	SLR	2005–2016	RQ_1
Sousa et al. [10]	Monitoring & Tools	ATD	SMS	2012–2022	RQ_1
Koulla Moulla et al. [11]	TD Measurement	Code, Design, Arch	Exploratory LR	2010–2023	RQ_1
Perera et al. [12]	TD Quantification Approaches	Code, Design, Arch	SMS	to 2023	RQ_1
Khomyakov et al. [13]	Automated TD Measurement	Various	SLR	2011–2018	RQ_1
Klimczyk and Madeyski [14]	Estimation and TD Impact	Various	SMS	–	RQ_1
Alfayez et al. [15]	TD Prioritization	General	SLR	1992–2018	RQ_1
Lenarduzzi et al. [16]]	Prioritization Strategies	Code, Arch	SLR	to 2020	RQ_1
Perera et al. [17]	RTD Quantification	Requirements	SMS	2012–2022	RQ_1
Melo et al. [18]]	Requirements TD	RTD	SLR	2010–2020	RQ_1
Saraiva da Silva et al. [19]	TD Tool Support	All Types	SMS	to 2020	RQ_1
Kleinwaks et al. [20]	TD in Systems Engineering	Theoretical	SLR	–	RQ_1
Behutiye et al. [21]]	TD in Agile Practices	Code, Arch, Process	SLR	1992–2014	RQ_1
Villa et al. [22]	TD in Microservices	Code, Arch	SMS	2015–2020	RQ_1
Ribeiro et al. [23]	Repayment Decision Criteria	All Types	SMS	to 2014	RQ_1
Alves et al. [24]	Identification and TDM	All Types	SMS	2010–2014	RQ_1
Fernandez-Sanchez et al. [25]	Elements of TDM	General	SMS	to 2015	RQ_1
Li et al. [26]	TDM Activities & Tools	All Types	SMS	1992–2013	RQ_1
Ampatzoglou et al. [27]	Financial Aspects of TD	Economic Perspective	SLR	to 2013	RQ_1
Lunde and Colomo-Palacios [28]	TD in DevOps/CI/CD	Code, Infra	SLR	to 2019	RQ_1
Nielsen et al. [29]	TD in Digital Government	General	SLR	2017–2020	RQ_1
Becker et al. [30]	Trade-off Decisions	Process/Time	SLR	to 2014	RQ_1
Albuquerque et al. [31]	Intelligent Techniques for TDM	All Types	SMS	2012–2021	RQ_1, RQ_2
Bogner et al. [32]	TD in AI Systems	Data, Model, Ethics	SMS	to 2020	RQ_1, RQ_2
Saeeda et al. [33]	Non-Technical Debt (NTD)	Process, Org, Social	MLR	to 2022	RQ_2
Our Work	**Evolution of Technical Debt**	**All Types**	**MLR**	**to 2024**	**RQ_1, RQ_2**

software companies. The search process was performed in six Digital Libraries (DLs) (namely ACM, IEEE, ScienceDirect, SpringerLink, Scopus, and Web of Science) from 2005 to 2016. After applying the selection criteria, 43 studies were identified. As a result of the study, the authors proposed a model in which the model categorizes the main characteristics of ATD and elucidates their interrelations. The findings suggest that by utilizing this model, stakeholders can enhance system success rates and mitigate adverse outcomes by increasing awareness of ATD.

Sousa et al. [10] conducted a Systematic Mapping Study (SMS) to explore methods and tools for identifying, monitoring, and managing ATD. The search process was performed in four DLs (namely Scopus, Web of Science, IEEE, and ACM) from 2012 to 2022. After applying the selection criteria, 70 studies were identified. As a result of the study, the authors proposed a structured roadmap to assist software engineers in identifying and monitoring ATD items systematically.

Koulla Moulla et al. [11] performed an exploratory literature review to explore the current state of research on measuring technical debt. The authors executed a comprehensive search across five DLs: Scopus, ScienceDirect, ACM, IEEE, and SpringerLink between 2010 and 2023. At the end of the selection process, 21 primary studies were retained for further analysis. The results of the study suggest that TD measurement primarily focuses on code quality, design issues, and architectural concerns. Additionally, the authors identified (a) various measurement solutions for measuring TD (static analysis tools, code metrics, TD principal, TD interest, etc.), (b) TD categorization methods, and (c) the research gaps identified in TD measurement research (e.g., measuring other types of debt beyond code debt, quantifying risks, etc.).

Perera et al. [12] performed a systematic mapping study to explore and categorize existing approaches for quantifying Technical Debt in software systems, focusing on code, design, and architectural levels. The search process was conducted in four DLs (namely Scopus, ACM, IEEE, and SpringerLink), and identified 96 primary studies. Based on the results, the authors explored 39 distinct quantification approaches for TD. Additionally, they analyzed these approaches by classifying them based on a set of abstract TD Quantification concepts and their high-level themes, including process/time, cost, benefit, probability, and priority. Khomyakov et al. [13] executed a systematic literature review to investigate existing approaches for measuring and analyzing TD, focusing on quantitative methods that can be automated. The search strategy was performed

in three DLs: ACM, IEEE, and Google Scholar between 2011 and 2018. After applying filtering criteria, 21 papers were selected for in-depth analysis. The results suggest that in the literature, there are various methods to measure TD, each using different criteria and not necessarily building upon existing models, however, the field is not yet mature. Many models lack validation, and there is a scarcity of tools to automate TD assessment.

Klimczyk and Madeyski [14] conducted a systematic mapping study to identify estimation problems arising from previously introduced TD in software projects and to explore solutions that address these challenges. The search process was conducted in five DLs (namely Scopus, ACM, IEEE, SpringerDirect, and SpringerLink), identifying 42 primary studies. The study highlights the significant influence of TD on software project estimations and underscores the necessity for incorporating TD management into estimation practices. Addressing TD proactively can lead to more accurate project planning and resource allocation.

Alfayez et al. [15] performed a systematic literature review on technical debt prioritization. The search was constrained to published literature from 1992 to 2018 in the DLs: ACM, Google Scholar, IEEE, Inspec, ScienceDirect, Scopus, Springer, and Web of Science. Additionally, they search manually in two TD conferences. At the end of the selection process, 23 primary studies were retained for further analysis. The study identified 24 TD prioritization approaches, which employed the use of 10 prioritization techniques.

Additionally, Lenarduzzi et al. [16] conducted a systematic literature review to investigate existing approaches for prioritizing TD in software engineering, focusing on the strategies, processes, factors, and tools utilized in TD prioritization. The authors executed a comprehensive search across eight DLs (namely ACM, IEEE, ScienceDirect, Scopus, Google Scholar, CiteSeer, Inspec, and Springerlink) and on the most important conferences and workshops on TD until 2020. At the end of the selection process, 44 primary studies were retained for further analysis. The results of the study suggest that code and architectural debt are the most frequently investigated types of debt when considering TD prioritization. Moreover, the study provides an impact map that highlights 53 factors related to the impact of TD to be considered for prioritization.

Perera et al. [17] conducted a systematic mapping study to identify and analyze existing Requirements Technical Debt (RTD) quantification approaches. The search strategy was conducted in four digital libraries (ACM, IEEE, Scopus, and ScienceDirect) between 2012 and 2022. After the selection criteria, 7 authors were selected for 7 studies. Then, the authors developed a conceptual model, namely RTD Quantification Model (RTDQM), to represent and compare various RTD quantification approaches through a common framework. Furthermore, Melo et al. [18] executed a systematic literature review to identify the state of the art related to TD management in software requirements, focusing on identification and measurement. The search process was conducted between 2010 and 2020 in five DLs (namely Scopus, ACM, ScineceDirect, IEEE, and SpringerLink), identifying 61 primary studies. The study identified several causes related to RTD, including strategic decisions for immediate gains and unintended changes in context. However, the authors explored that the research on RTD is still in its early stages, especially concerning management tools, whereas there is a need for metrics to support the measurement stage of RTD.

Saraiva da Silva et al. [19] conducted a systematic mapping study to identify and analyze available tools for managing TD. They focused on understanding the activities, functionalities, and types of TD addressed by these tools. The search process was applied on six DLs (namely: ResearchGate, ACM, IEEE, ScienceDirect, and Scopus) until January 2020. At the end of the selection process, 47 primary studies were retained for further analysis, and 50 tools. As an outcome of this research is a holistic view of TD tools regarding the features proposed by them to address TD in different dimensions and a categorization that describes and encompasses the main characteristics of the tools.

Kleinwaks et al. [20] performed a systematic literature review to investigate how the concept of TD is applied within the field of systems engineering, aiming to identify its usage and the stages of the systems engineering lifecycle most susceptible to technical debt. The search strategy was conducted on four DLs, namely IEEE, ScienceDirect, Wiley, and SpringerLink, selecting 18 relevant studies for detailed analysis. The results of the study show that the TD metaphor is not widely prevalent in systems engineering research, and the existing research primarily focuses on theoretical aspects rather than practical applications.

Behutiye et al. [21] conducted a systematic literature review to analyze and synthesize the current state of knowledge regarding TD in agile software development, focusing on its causes, consequences, and management strategies. The search strategy was conducted in six DLs (ACM, Google Scholar, IEEE, ProQuest, Scopus, and the Web of Science) between 1992 and June 2014. After the selection criteria, the authors selected 38 studies.

The results of the study identified five key research areas related to TD in agile, with the highest attention given to managing TD, followed by architecture and its relationship with Additionally, they found eight categories of causes for incurring TD in agile, focusing on quick delivery, whereas five categories of consequences were identified, including reduced productivity, system degradation, and increased maintenance costs. Finally, the authors explored twelve strategies for managing TD in agile, with refactoring and enhancing the visibility of TD being the most significant.

Villa et al. [22] conducted a systematic mapping study to gather evidence and characterize technical debt in systems with microservices architecture. The search strategy was performed in four DLs: ACM, IEEE, ScienceDirect, and SpringerLink between 2015 and 2020. After applying filtering criteria, 12 papers were selected for in-depth analysis. The results of the study identified eight types of technical debt, with architecture debt and code debt being the most frequently reported. Additionally, the study identified that design and implementation activities are those with the highest number of cases of

technical debt, followed by migration activities.

Ribeiro et al. [23] performed a systematic mapping study to identify decision-making criteria related to TD repayment in software projects. The search was constrained until 2014 in three DLs: ACM, IEEE, and Scopus. At the end of the selection process, 38 primary studies were retained for further analysis. The results of the study identified a list of 14 criteria that can be used to support decision-making on the repayment of TD items, categorized them into four categories (i.e., nature of the TD, customer, effort, and project). These criteria include factors such as the severity of the debt, the existence of workarounds, the visibility, the impact on customers, the impact on the project, the scope of tests, the nature of the project, and the lifetime of the system. Additionally, the study provided a list of types of TD related to the identified decision-making criteria, helping software engineers to understand which criteria apply to specific TD types.

Alves et al. [24] conducted a systematic mapping study on identification and management of TD. The search strategy was performed in 8 DLs (namely ACM, IEEE, ScienceDirect, Engineering Village, SpringerLink, Scopus, Citeseer, and DBLP) from 2010 to 2014. After the selection criteria, the authors selected 100 studies.

The authors proposed: (a) a taxonomy of TD types; (b) a list of proposed TD management strategies; (c) an analysis of the types of empirical evaluation performed on the studies; (d) a list of data sources used in TD identification activities; and (e) a list of software visualization techniques used to identify and manage TD.

Additionally, Fernandez-Sanchez et al. [25] performed a systematic mapping study to identify elements that are required to manage TD. The search strategy was performed in six DLs (IEEE, ACM, Scopus, ScienceDirect, Web of Science, and SpringerLink) until 2015. After the selection criteria, the authors selected 63 studies.

The study identified the elements of technical debt management and the methods that support the elements in the industrial environment. Moreover, the authors classified the elements into three groups (basic decision-making factors, cost estimation techniques, practices, and techniques for decision-making) and mapped them according to three stakeholders' points of view (engineering, engineering management, and business-organizational management).

Li et al. [26] conducted a systematic mapping study, which was aimed at collecting studies on TD and TDM to identify and classify activities employed for managing TD. The search strategy was performed in seven DLs (namely IEEE, ACM, Scopus, ScienceDirect, Web of Science, Inspec, and SpringerLink) between 1992 and 2013. As a result, 94 studies were found for further analysis. The study classified TD into 10 types, identified 8 TDM activities, and 29 tools for TDM.

Ampatzoglou et al. [27] performed a systematic literature review to analyze the financial aspects of managing technical debt. The search was conducted until 2013 in seven DLs: ACM, Google Scholar, IEEE, SpringerLink, ScienceDirect, Scopus, and Web of Science. At the end of the selection process, 69 primary studies were retained for further analysis. The results of the study suggest that the most common financial terms that are used in TD research are principal and interest, whereas the financial approaches that have been more frequently applied for managing technical debt are real options, portfolio management, cost/benefit analysis, and value-based analysis.

Lunde and Colomo-Palacios [28] conducted a systematic literature review to explore the relationship between continuous software engineering practices (such as continuous integration, delivery, deployment, and DevOps) and the accumulation of technical debt. The search strategy was performed in four DLs: ACM, IEEE, ScienceDirect, and SpringerLink until 2019. After applying filtering criteria, 23 papers were selected for in-depth analysis. The results suggest that while continuous practices aim to enhance software quality and delivery speed, they can also contribute to the accumulation of technical debt if not managed properly.

Nielsen et al. [29] conducted a systematic literature review on TDM for digital government. The search strategy was conducted in 6 DLs (ACM, Google Scholar, IEEE, ProQuest, Scopus, and the Web of Science) between 2017 and 2020. After the selection criteria, the authors selected 49 studies.

The authors identified several gaps in the existing literature, such as an absence of theory explaining the relation of events, a shortage of studies conducted in the public sector, and an absence of specific techniques to study the actual technical debt management behavior. The authors proposed a research agenda for digital government scholars to address these gaps.

Becker et al. [30] performed a systematic literature review to examine how trade-off decisions related to TD are studied in software engineering. The authors focus on time-based trade-offs, i.e., how short-term benefits and long-term costs are balanced in TD management. The search was constrained until 2014 in three data sources: ACM, IEEE, and Scopus. At the end of the selection process, there are only 9 studies that explicitly used empirical methods to study specific trade-off decisions. The results explored that while many studies emphasize engineering measures and idealized decision-making processes, few have investigated how decisions are made in practice.

Albuquerque et al. [31] performed a systematic mapping study on managing TD using intelligent techniques. The search strategy was performed in four data sources: ACM, IEEE, Scopus, and Engineering Village from 2012 to 2021. After the selection criteria, the authors selected 150 studies.

The results of the study identified a variety of intelligent techniques employed in TDM, including machine learning, data mining, and artificial intelligence. Additionally, the authors emphasized the need to expand research beyond code-level TD to address other forms, such as architectural and design debt.

Bogner et al. [32] conducted a systematic mapping study to provide an overview and characterization of the types of TD and antipatterns in AI systems. The search strategy was conducted in Google Scholar until mid-2020, selecting 21 relevant studies for detailed analysis. The results of the study suggest that there are 4 new TD types, namely data, model, configuration, and ethics debt, in AI-based systems. Additionally, the

authors identified 72 antipatterns, most of which are related to data and model deficiencies, and 46 solutions that are either to address specific TD types, antipatterns, or TD in general.

Finally, Saeeda et al. [33] conducted a multivocal literature review on non-TD (NTD) in software development. The search process was performed until October 2022 in the Google Scholar engine, retrieving 40 papers for in-depth analysis. The results of the study identified 5 NTD types, namely process debt, people debt, social debt, cultural debt, and organizational debt. Additionally, the study explored the causes of NTD (e.g., poor customer responsiveness, Insufficient communication, weak organizational culture, etc.) and potential mitigation strategies (e.g., measurement of process, continued monitoring and communication, creating the right mindset in the company, etc.).

While prior studies have examined particular aspects of TD in AI systems, such as model versioning, data quality, or architecture-level concerns, our work provides a more comprehensive and updated perspective by analyzing the evolution of the concept and management of TD in response to the adoption of ML and GenAI. Furthermore, contrary to previous reviews that exclusively focused on ML or on classical topic modeling TD categories, we adopt a more comprehensive approach that incorporates academic and gray literature to capture emerging industry practices. Building on these concepts, our study employs a dual-perspective approach, anchored by two complementary themes: adapting software engineering practices for AI (SE4AI) and understanding how AI alters the software lifecycle (AI4SE). This dual lens reflects the bidirectional influence between AI technologies and TD, offering a comprehensive and nuanced understanding of the relationship. This positioning ensures that our work is relevant for researchers and practitioners seeking to understand and navigate the new forms of debt introduced, initially in ML and now in the GenAI era.

3. Methodology

To understand the state of the art and the practice of Technical Debt prioritization, we conducted a multivocal literature review based on the guidelines defined by Garousi et al. [34]. In this Section, we describe the goal and the research questions (Section 3.1) and report our search strategy approach (Section 3.2). Moreover, we performed a quality assessment (Section 3.3) for each included paper and outlined the data extraction and the analysis (Section 3.4) of the corresponding data.

3.1. Goal and Research Questions

Our main goal is to explore the evolution of TD in the context of the changing landscape of software development and operations. Specifically, we want to examine how TD has evolved alongside the transition from traditional DevOps to MLOps and, more recently, to GenAIOps.

Based on our goal, we defined two research questions (RQ_s):

RQ_1

How does TDM need to be tailored to match the needs of developing AI-driven software?

By addressing $\mathbf{RQ_1}$, we aim to identify the specific adaptations required in TDM strategies and tools to effectively manage debt in today's evolving AI landscape. This includes understanding how existing methodologies may need to be restructured, what new techniques or automation capabilities are necessary, and how organizations can proactively mitigate Technical Debt in AI-centric development workflows.

RQ_2

How does the use of AI during code development affect TDM?

By answering $\mathbf{RQ_2}$, we aim to provide a balanced perspective on how AI influences TDM, highlighting both its advantages and the unintended consequences that organizations must consider when leveraging AI for software development. As an example, AI has the potential to enhance TDM in several ways. First, it can automate code refactoring, optimize maintenance efforts, predict debt accumulation, and improve decision-making through intelligent recommendations. However, it's important to note that AI itself introduces new forms of Technical Debt, such as model drift, data dependencies, and explainability concerns. These could complicate software maintenance and long-term sustainability.

3.2. Search Strategy

The search strategy involves the outline of the most relevant bibliographic sources and search terms, the definition of the inclusion and exclusion criteria, and the selection process relevant to the inclusion decision. Our search strategy is depicted in Figure 2.

Search terms. When constructing the search string, we organized it into three main components. First, we aimed to explicitly include the most commonly studied types of Technical Debt, as identified in prior literature, to ensure we captured works related to debt prioritization and categorization. Second, we incorporated the term "debt". Third, we aligned the query with the overall objective of our review by incorporating terms related to Machine Learning and its associated paradigms (e.g., Deep Learning, Generative AI). This layered approach allowed us to systematically retrieve both focused and peripheral studies addressing the evolving relationship between TD and AI-driven software development. The developed search string contained the following search terms:

(Technical OR design OR architect* OR test OR implementation OR documentation OR requirement OR code OR Infrastructure OR versioning OR defect OR build OR testing)

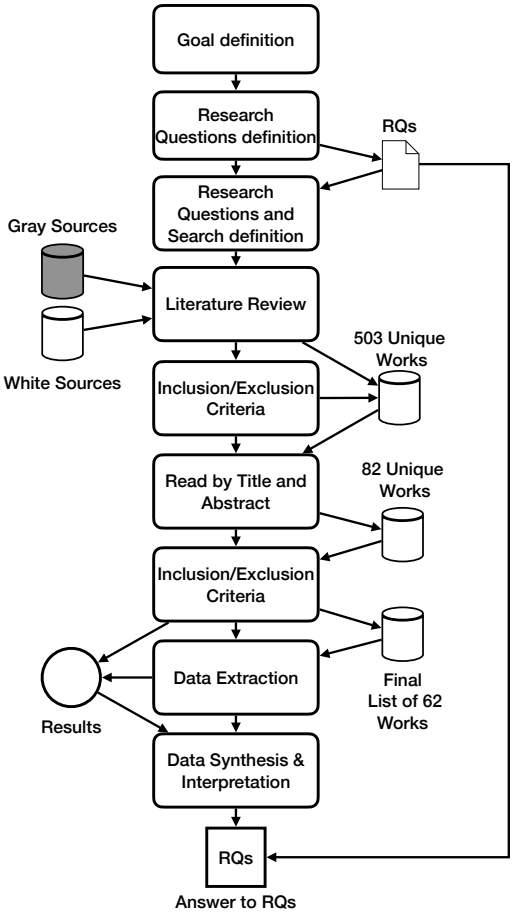

Figure 2: The Search and Selection Process

AND
Debt
AND
(GenerativeAI OR "Generative AI" OR "Machine Learning" OR MLOps OR LLMOps OR SLM OR GenAI OR "Gen AI" OR SLMOps OR "large language model*" OR "small language model*" OR GPT* OR ChatGPT OR LLM OR Gemini OR Llama OR ML OR GenAIOps)

Bibliographic sources. We selected the list of relevant bibliographic sources for peer-reviewed literature following the suggestions of Kitchenham and Charters [35], since these sources are recognized as the most representative in the software engineering domain and used in many reviews. The list for the white literature includes: *ACM Digital Library, IEEEXplore Digital Library, Web of Science, and Scopus.* For the grey literature (e.g., non-peer-reviewed), we included the first 200 entries from each search from Google, Yahoo, and Bing. This limit was set to balance breadth and relevance, as general-purpose search engines tend to produce redundant results, often favoring mirrored content and infinite scrolling. We found that extending beyond 200 entries added little value, as many hits were duplicates or low-quality. Duplicates were removed during screening, after which we applied our inclusion criteria.

Inclusion and exclusion criteria. We defined inclusion and exclusion criteria to be applied to the title and abstract (T/A), the full text (F), or both cases (All), as reported in Table 2.

Table 2: Inclusion and exclusion criteria

Criteria	Assessment Criteria	Step
Inclusion	Papers discussing TD management have changed when developing ML or GenAI applications	All
	Papers discussing how ML and GenAI changed the software development lifecycle concerning TD	All
Exclusion	Papers not fully written in English	T/A
	Duplicated papers and Books	T/A
	Out of topic	T/A
	Non peer-reviewed papers	T/A
	Not accessible by institution	T/A
	Position papers/discussion papers that present a new method without validation	T/A
	Videos	T/A
	Paper about **Generative AI** dated before 2021	T/A

Search and selection process. The search was conducted in September 2024 and included all the publications available until this period. The application of the search terms returned 894 unique works, composed of **433 unique peer-reviewed papers and 461 online entries**.

Testing the applicability of inclusion and exclusion criteria: Before applying the inclusion and exclusion criteria, we tested their applicability [36] on a subset of 10 papers (assigned to two authors) randomly selected from the papers retrieved.

Applying inclusion and exclusion criteria to title and abstract: We applied the refined criteria to the remaining 894 works. Each work was read by two authors; in the case of disagreement, a third author was involved in the discussion to clear up any such disagreement. For 27 works, we involved a third author. Out of the **894** initial papers, we included **82** based on title and abstract. The application of the inclusion and exclusion criteria resulted in an almost perfect agreement (Cohen's Kappa coefficient = **0.8652**).

Full reading: We fully read the 82 papers included by title and abstract, applying the same criteria defined in Table 2 and assigning each one to two authors. We involved a third author for 6 papers to reach a final decision. Based on this step, we discarded 20 papers and therefore, selected **62** papers as possibly relevant contributions, as reported in Table 4. The application of the inclusion and exclusion criteria resulted in a substantial agreement (Cohen's Kappa coefficient = **0.7944**).

3.3. Quality Assessment

Before proceeding with the review, we checked whether the quality of the selected papers was sufficient to support our goal and whether the quality of each paper reached a certain quality level. We performed this step according to the protocol proposed by Dybå and Dingsøyr [37]. To evaluate the selected

papers, we prepared a checklist (Table 3) with a set of specific questions. We ranked each answer, assigning a score on a five-point Likert scale (0=poor, 4=excellent). A paper satisfied the quality assessment criteria if it achieved a rating higher than (or equal to) 2. To account for the different nature of white and grey literature while maintaining methodological consistency, we adopted a differentiated quality assessment strategy grounded in the same protocol. While all sources were evaluated using this common framework, a more flexible approach was applied to the grey literature. Given that the concepts explored, particularly the relationship between DataOps and TD in the GenAI era, are still emerging and only partially addressed in peer-reviewed work, we selected a subset of criteria from the checklist for non-academic sources. Specifically, we applied criteria 2, 3, 10, and 11 (in bold in Table 3), which assess the clarity of the stated aim, contextual grounding, relevance of contributions, and practical applicability. This selective strategy ensured that the included grey literature maintained both relevance and credibility in relation to the study's objectives.

Table 3: Quality Assessment Criteria

QA$_s$	**Quality Assessment Criteria (QA)**
QA$_1$	Is the paper based on research (or is it merely a "lessons learned" report based on expert opinion)?
QA$_2$	**Is there a clear statement of the aims of the research?**
QA$_3$	**Is there an adequate description of the context in which the research was carried out?**
QA$_4$	Was the research design appropriate to address the aims of the research?
QA$_5$	Was the recruitment strategy appropriate for the aims of the research?
QA$_6$	Was there a control group with which to compare treatments?
QA$_7$	Was the data collected in a way that addressed the research issue?
QA$_8$	Was the data analysis sufficiently rigorous?
QA$_9$	Has the relationship between researcher and participants been considered to an adequate degree?
QA$_{10}$	**Is there a clear statement of findings?**
QA$_{11}$	**Is the study of value for research or practice?**

Response scale: 4 (Excellent), 3 (Very Good), 2 (Good), 1 (Fair), 0 (Poor)

Among the 894 papers included in the review from the search and selection process, only **62** were not discareded in the previous steps and also fulfilled the quality assessment criteria, as reported in Table 4.

Table 4: Results of search and selection, and application of quality assessment criteria

Step	**# Papers**
Retrieval from bibliographic sources (unique)	894
Reading by title and abstract	-812
Full reading	-20
Quality Assessment Criteria	**-0**
Primary Studies	**62**

3.4. Data Extraction

We extracted data from the Primary Studies (PSs) that satisfied the quality assessment criteria and categorized them into two main categories. The first category includes studies addressing the question: *"How has Technical Debt (TD) management changed when developing ML or GenAI applications?"*—corresponding to **RQ**$_1$. The second category includes studies focusing on *"How ML and GenAI have influenced the software development lifecycle about TD"*—corresponding to **RQ**$_2$. Following this initial classification, we defined the specific data to be extracted for each category. The data extraction form, along with the mapping of the information needed to answer each research question, is summarized in Table 5. For RQ$_1$, we began by identifying the study's main ~~goal~~ theme to understand its context. We then extracted the types of TD management activities described, as well as the types of TD considered. In both categories, we identified the challenges reported to highlight areas requiring further investigation.

For RQ$_2$, in addition to the challenges, we also extracted the opportunities discussed in the studies. Furthermore, we focused on two key aspects: the impact of the work and the main problem it aimed to address.

Table 5: Data Extraction

RQ$_s$	**Data**	**Outcome**
RQ$_1$	~~Goals~~Themes	Study ~~Goal~~Theme
	Activities	Activities of TD management
	Type	Type of TD identified in the work
	Challenges	Challenges faced in the work
RQ$_2$	Impact	What is the impact of the work
	Problem	Main problem addressed
	Challenges	Challenges faced in the work
	Opportunities	Opportunities encountered in the work

3.5. Replicability

To allow replication and extension of our work by other researchers, we prepared a replication package[6] for this study with the complete results obtained.

4. Results

In this section we present the results to answer our RQs. Each study was first categorized based on its alignment with one of two conceptual perspectives, which we use as a shorthand to map to our research questions.

Studies addressing RQ$_1$ ("How has Technical Debt (TD) management changed when developing ML or GenAI applications?") are framed as focusing on *adapting to emerging software*", while studies aligned with RQ$_2$ ("How ML and GenAI have influenced the software development lifecycle about TD") reflect the perspective of *software evolving as a result*

[6] https://doi.org/10.6084/m9.figshare.28930268

Table 6: White and gray literature distribution

Code	PSs	#
Conference Paper (White)	[PS1], [PS2], [PS3], [PS4], [PS5], [PS6], [PS7], [PS8], [PS9], [PS10], [PS11], [PS12], [PS14], [PS15], [PS16], [PS17], [PS18], [PS19]	18
Journal Article (White)	[PS13]	1
Grey Sources	[PS20], [PS21], [PS22], [PS23], [PS24], [PS25], [PS26], [PS27], [PS28], [PS29], [PS30], [PS31], [PS32], [PS33], [PS34], [PS35], [PS36], [PS37], [PS38], [PS39], [PS40], [PS41], [PS42], [PS43], [PS44], [PS45], [PS46], [PS47], [PS48], [PS49], [PS50], [PS51], [PS52], [PS53], [PS54], [PS55], [PS56], [PS57], [PS58], [PS59], [PS60], [PS61], [PS62]	43

Table 7: RQ-baseds distribution

RQs	PSs	#
RQ1	[PS1], [PS2], [PS3], [PS4], [PS6], [PS7], [PS8], [PS9], [PS10], [PS11], [PS12], [PS13], [PS14], [PS15], [PS16], [PS17], [PS18], [PS19], [PS20], [PS23], [PS25], [PS26], [PS28], [PS32], [PS35], [PS36], [PS37], [PS41], [PS42], [PS43], [PS44], [PS45], [PS46], [PS48], [PS50], [PS52], [PS53], [PS54], [PS55], [PS56], [PS57], [PS60], [PS61], [PS62]	44
RQ2	[PS5], [PS21], [PS22], [PS24], [PS27], [PS29], [PS30], [PS31], [PS33], [PS34], [PS38], [PS39], [PS40], [PS47], [PS49], [PS51], [PS58], [PS59]	18

of its usage". These labels are not substitutes for the formal research questions but serve to clarify the thematic focus during categorization and discussion.

Before delving into the specifics of each RQ, we first analyzed the overview of the publication years and the countries of affiliation of the primary studies.

4.1. Preliminary analysis on the number of Primary Studies.

The final pool of 62 work extracted from the described methodology can be categorized in multiple ways.

The first main distinction we need to address is the one between works being part of white literature (or peer-reviewed) and those being part of the grey literature, described in Table 6. In our selected primary studies (PSs), we included 19 works from white literature (composed of 18 conference papers and 1 journal article) and 43 works from grey literature. This means that almost 70% of the PSs are from grey literature. Another way to categorize is by the main research question each PS addresses and is depicted in Table 7. In this classification we can assign 44 PSs to the first group and 18 PSs related to the second, which means that according to this classification almost 71% of the works are related to the first RQ.

The collected data shows a clearly increasing trend in academic and industry activity around the topic, showing the temporal evolution of attention to TD in the context of AI and ML (Figure 3).

The fact that some PSs included in the analysis originate from non-peer-reviewed sources has made it difficult to accurately trace their publication year. As a result, these studies

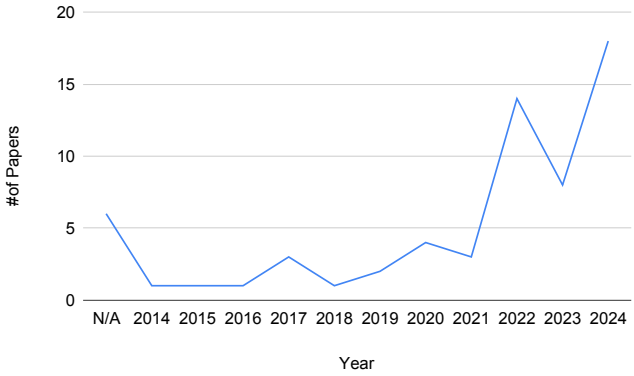

Figure 3: Number of PSs per year

are labeled as "N/A" in the figure. Despite this limitation, the overall trend observed in the remaining PSs provides a strong indication of how this topic has gained prominence over the years.

It is worth noting that the release of generative AI tools like ChatGPT in 2022 has led to a significant rise in interest. The rapid adoption and integration of these technologies has not only sped up the deployment of AI-driven applications but also raised concerns about the long-term sustainability, maintainability, and operational risks associated with ML systems. This heightened awareness has led to a surge in discussions surrounding TD specific to ML, including issues related to model degradation, data dependencies, ethical considerations, and the complexities of continuous monitoring and updating of AI systems.

4.2. How does TDM need to be tailored to match the needs of developing AI-driven software (RQ_1)

To answer RQ_1, we extracted 4 different factors: Study ~~Goal~~ Theme, TD management activities, TD types, and Challenges faced in the work (Table 5).

The analysis revealed a predominant focus on a particular type of TD, known as "TD specific to ML". This trend highlights the growing recognition of **ML-specific TD** [PS12], [PS17], [PS19], [PS20], [PS23], [PS32], [PS37], [PS42], [PS44], [PS45], [PS46], [PS48], [PS50], [PS52], [PS53], [PS54], [PS56], [PS57], [PS60], [PS61], [PS62] as a distinct and pressing concern, prompting researchers to explore new methodologies for identifying, understanding, and mitigating its unique challenges. Unlike traditional software engineering TD, ML TD introduces complexities related to model performance degradation, data dependencies, and the dynamic nature of learning systems (Figure 4).

A particularly noteworthy trend within this field is the emphasis on data-related TD. Nine studies [PS17], [PS21], [PS32], [PS42], [PS45], [PS46], [PS48], [PS54], [PS61] have specifically addressed **data debt**, emphasizing the critical role of data in the ML lifecycle and its direct impact on TD accumulation in AI systems. Unlike traditional TD, which is primarily associated with code and architectural decisions, ML TD is inher-

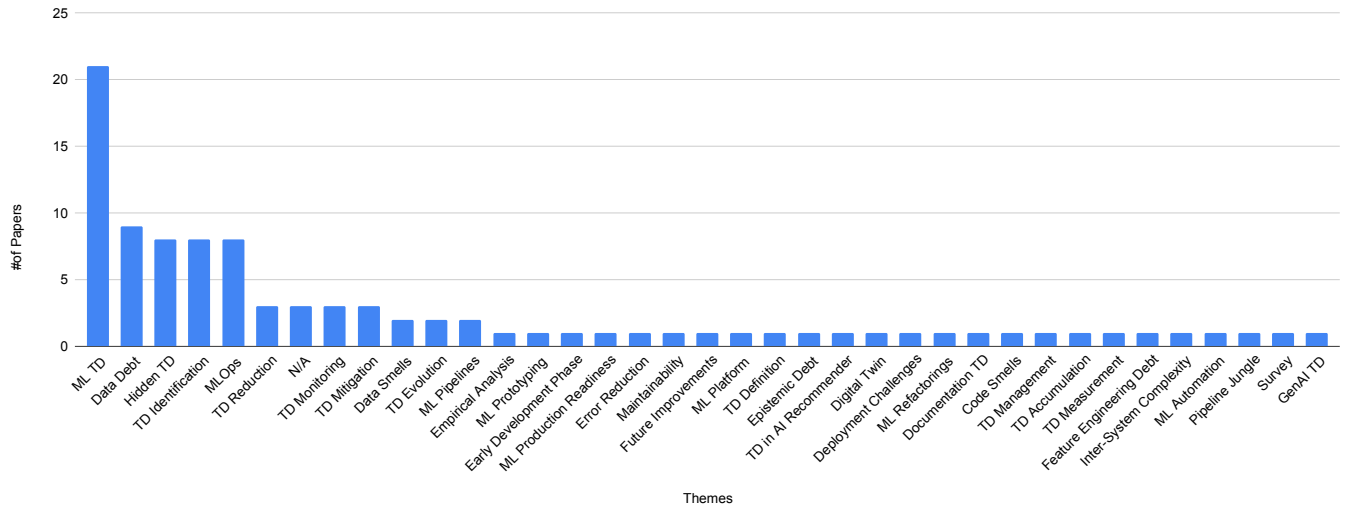

Figure 4: Study Themes (RQ_1)

ently linked to data quality, availability, and its evolution over time.

A clear example of this relationship is seen in the concepts of data drift and concept drift, which, much like traditional TD, describe the gradual degradation of system performance over time. However, in the case of ML, this degradation is driven by the strong coupling between models and their underlying data distributions. As data shifts—whether due to external factors, user behavior changes, or evolving real-world conditions—ML models can become outdated or misaligned. This necessitates frequent updates, retraining, or redesign efforts to maintain optimal performance.

These insights confirm that ML-specific TD is distinct from traditional software debt and necessitates specialized strategies for monitoring, managing, and mitigating its impact. The prevalence of data-related objectives within the analyzed studies further indicates that addressing ML TD requires a multidisciplinary approach, incorporating best practices in data engineering, continuous monitoring, and model lifecycle management to ensure the long-term sustainability and reliability of AI-driven systems. Eight PSs [PS10], [PS21], [PS24], [PS41], [PS50], [PS52], [PS53], [PS56] specifically address MLOps, reflecting a growing interest in operationalizing machine learning while systematically tackling its TD.

This finding indicates that researchers and practitioners are increasingly viewing **MLOps** as a crucial discipline for mitigating the risks associated with the deployment of ML models. These risks include the need to automate model retraining, ensure reproducibility, and integrate continuous monitoring mechanisms. This will help to manage evolving data and model performance over time. The convergence of TD in ML and MLOps research indicates a shift toward more structured and scalable approaches to sustaining AI-driven systems in production environments.

Figure 5 provides a comprehensive overview of the various TD management activities identified in the PSs analyzed. As

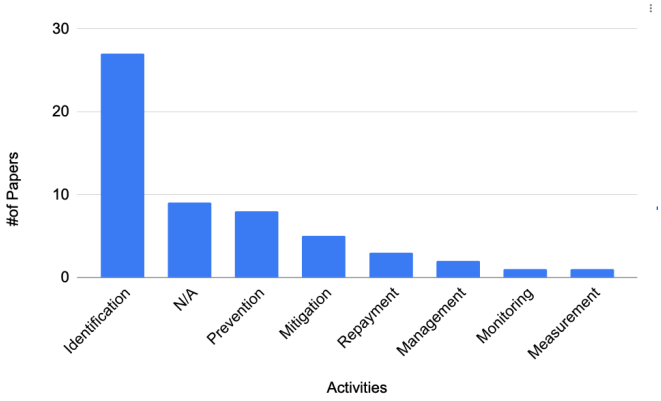

Figure 5: RQ_1: Activities of different PSs

illustrated, the activities extend across the entire range of TD management, from initial identification to proactive prevention and subsequent mitigation. However, it should be noted that for several studies [PS3], [PS24], [PS26], [PS32], [PS37], [PS56], [PS57], [PS62] a specific focus on TD management activities could not be determined based on the available information.

The most significant finding from the Figure is that the **identification of TD** [PS1], [PS2], [PS6], [PS7], [PS8], [PS10], [PS11], [PS13], [PS14], [PS15], [PS16], [PS18], [PS19], [PS20], [PS21], [PS23], [PS25], [PS35], [PS42], [PS43], [PS44], [PS46], [PS48], [PS50], [PS52], [PS54], [PS60] emerges as the predominant activity across the analyzed works, emphasizing a fundamental aspect of TD management. Recognizing and identifying TD instances is not only essential but also the most extensively explored area in the current literature. The increased focus on identification is a key response to the broader industry and academic initiatives aimed at systematically detecting, categorizing, and documenting various forms of TD, particularly in the context of machine learning and AI-driven systems, where

complexity is increasing.

Following identification, the activities of **prevention** [PS4], [PS9], [PS10], [PS12], [PS17], [PS21], [PS36], [PS61] and **mitigation** [PS42], [PS43], [PS44], [PS46], [PS60] are the next most frequent areas of focus among the primary studies. The prominence of these two activities is also unsurprising, given their complementary role in effective TD management. Prevention strategies aim to proactively avoid the accumulation of TD through improved design decisions, rigorous development practices, and adherence to best practices, which are particularly crucial in rapidly evolving machine learning ecosystems. Mitigation, on the other hand, involves reactive measures to reduce the impact of already accumulated debt. These measures encompass techniques such as refactoring, model retraining, and data cleansing to maintain or restore system integrity and performance.

Figure 6 presents a detailed breakdown of the various types of TD identified across the PSs. It is important to note that for certain studies [PS2], [PS9], [PS32], [PS36], [PS41], [PS43], [PS52], [PS53], [PS56], [PS57], and [PS62] we were unable to classify the specific type of TD due to insufficient information or unclear definitions provided within these works.

The data presented indicates that **infrastructure debt** [PS1], [PS4], [PS7], [PS8], [PS10], [PS11], [PS12], [PS13], [PS15], [PS16], [PS17], [PS18], [PS23], [PS26], [PS35], [PS36], [PS42], [PS44], [PS45], [PS46], [PS48], [PS50], [PS61] is the most frequently identified type of debt, with 23 separate studies addressing it. This prevalence highlights a significant challenge in managing the underlying hardware, cloud resources, deployment pipelines, and continuous integration/continuous deployment (CI/CD) environments essential for supporting robust ML systems. In the context of ML, common challenges often include suboptimal resource management, lack of scalability, and inadequate maintenance of deployment environments. These issues can result in performance bottlenecks, downtime, and increased operational costs.

The second most prevalent TD type identified is **code debt** [PS3], [PS12], [PS13], [PS16], [PS18], [PS19], [PS23], [PS24], [PS25], [PS42], [PS44], [PS45], [PS46], [PS48], [PS50], [PS60] as indicated in 16 studies. In the context of ML, the rapid experimentation and iterative development practices towards higher performance of employed models, can swiftly result in the accumulation of code-related debt. Over time, this debt can impede debugging, refactoring, and the integration of new features or models, hindering the overall maintainability of the system and its evolution.

Interestingly, both architectural debt and data debt are identified as the third most frequent types, with each observed in 14 studies. **Architectural debt** [PS7], [PS8], [PS10], [PS11], [PS12], [PS15], [PS18], [PS20], [PS25], [PS42], [PS45], [PS46], [PS50], [PS61] differs from infrastructure debt in that it concerns the high-level system design and structure, including decisions related to modularity, interoperability, and adaptability of the overall system architecture. Poor architectural choices in ML systems can result in rigid dependencies, limit scalability, and make it difficult to integrate new algorithms or technologies.

The prevalence of **data debt** [PS6], [PS8], [PS10], [PS12], [PS14], [PS15], [PS16], [PS17], [PS21], [PS25], [PS35], [PS48], [PS54], [PS60] in the context of ML and GenAI is not unexpected, given the central role that data plays in defining system performance. Data debt encompasses a range of issues, including inadequate data quality, outdated datasets, inadequate data governance, and inefficiencies in data pipelines. These challenges are particularly salient in these fields, as suboptimal or mismanaged data can significantly compromise model performance, introduce bias, and necessitate costly retraining efforts.

Figure 7 presents an overview of the key challenges associated with TD in the analyzed PSs. Notably, a significant portion of studies did not explicitly identify specific challenges [PS2], [PS4], [PS8], [PS11], [PS14], [PS16], [PS17], [PS18], [PS20], [PS23], [PS24], [PS26], [PS36], [PS41], [PS45], [PS48], [PS52], [PS53], [PS57], [PS62]. This dominance of studies without identified challenges suggests a potential gap in the literature, indicating that while TD is widely discussed, its specific challenges may not always be systematically categorized or explicitly addressed.

Among the challenges that have been identified, issues related to **ML pipelines and feedback loops** [PS1], [PS6], [PS9], [PS10], [PS12], [PS13], [PS15], [PS19], [PS21], [PS25], [PS32], [PS37], [PS42], [PS46], [PS50], [PS54], [PS56],[PS60],[PS61] are the most frequently mentioned, identified in 19 studies. This trend aligns with the growing interest in MLOps, which has emphasized the need for automated pipelines and structured feedback loops to enhance model development, deployment, and maintenance. The rising adoption of these systems is also contributing to the accumulation of TDs as automation mechanisms introduce new complexities and maintenance burdens. Ensuring the effective functioning of feedback loops without introducing unintended biases or reinforcing outdated models has become a key concern within the ML community.

Following, **technical and architectural debt** [PS7], [PS10], [PS12], [PS13], [PS15], [PS21], [PS25], [PS35], [PS37], [PS44], [PS46], [PS56], [PS60], [PS61] remain prominent concerns. These issues primarily arise from the rapid evolution of ML frameworks, the challenge of maintaining scalable architectures, and the accumulation of legacy code that is often not designed with long-term maintainability in mind. The complexity of AI-driven software systems further increases architectural TD, as dependencies between components, model retraining workflows, and integration with evolving data sources require continuous adaptations and refinements. This is a particularly relevant point in ML-based architectures as most of the dependencies between components rely on external libraries. Such external libraries are developed and delivered by entities external to the development team, making the team responsible on assessing the risk of abandonment of such tools before adopting them in their project [38]. It is interesting to note that **data dependencies and evolution** [PS6], [PS9], [PS10], [PS12], [PS15], [PS21], [PS32], [PS42], [PS50], [PS56], [PS61] do not rank among the top three challenges, despite the well-established centrality of data in ML and GenAI applications. Given the critical role that data quality, availability,

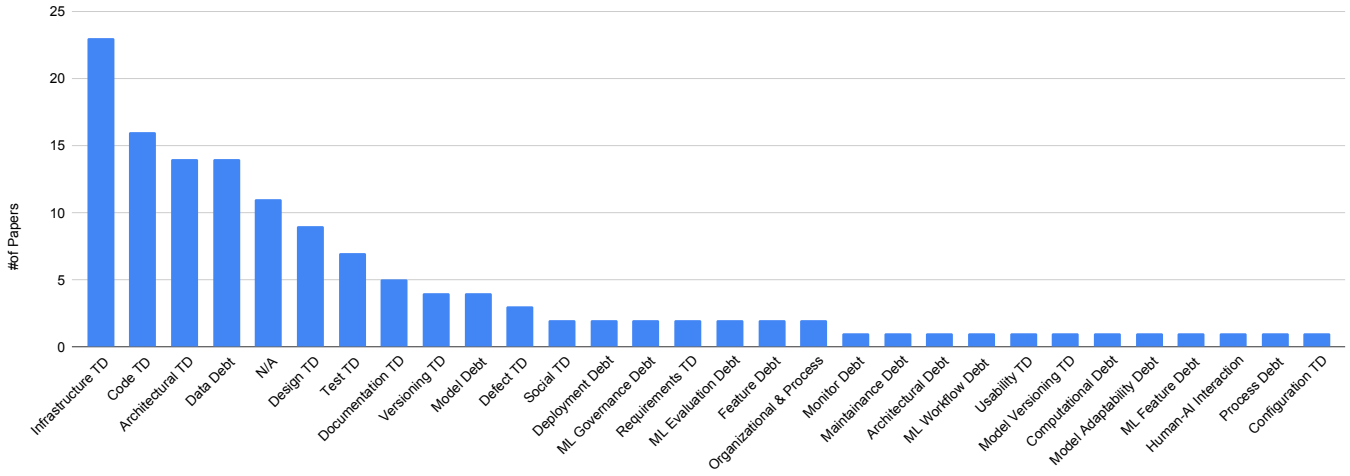

Figure 6: RQ_1: TD Types of different PS

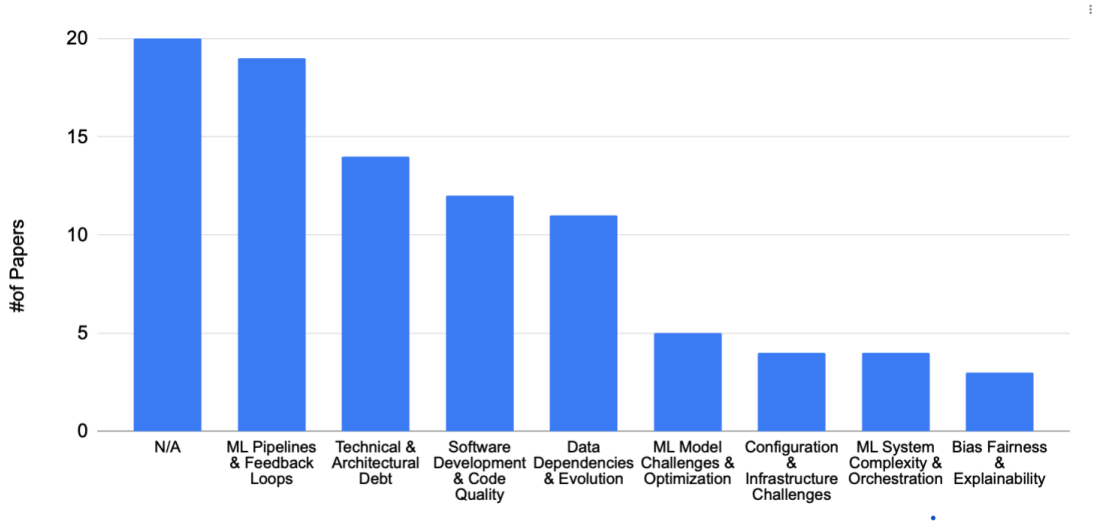

Figure 7: RQ_1: Challenges of different PSs

and consistency play in model performance and sustainability, one would expect data-related challenges to be among the most frequently discussed. This discrepancy may be indicative of a twofold phenomenon: first, that data issues are widely acknowledged; second, that they are often considered inherent to ML workflows rather than explicitly framed as a form of TD. Alternatively, this lack of emphasis could indicate a need for further research into how evolving data dependencies contribute to long-term ML system maintainability and TD accumulation.

4.3. AI usage affect on TDM during code development (RQ_2)

To answer RQ_2, we extracted 3 different factors.

Figure 8 provides a detailed overview of the various impact categories that have been investigated in the PSs. It is important to note that in this case only for one work was not possible to identify the impact [PS55].

The category with the highest mentions for impact is **TD Management & Reduction** [PS5], [PS24], [PS27], [PS29], [PS30], [PS31], [PS34], [PS40], [PS51], [PS55]. This underscores the increasing awareness and emphasis on addressing TD within AI and ML systems. The focus on managing and reducing TD indicates a shift towards more sustainable development practices, with the aim of enhancing long-term maintainability, scalability, and efficiency in AI-driven applications. The growing recognition of TD's impact highlights the need for structured frameworks and automation strategies to mitigate its effects.

The second most frequently identified impact category is **AI-Driven Code Automation & Management** [PS22], [PS24], [PS27], [PS29], [PS31], [PS34], [PS40], which highlights the growing influence of AI in automating various aspects of software engineering, from optimizing development practices to improving maintainability. AI-powered automation is being used more and more to streamline workflows, reduce manual

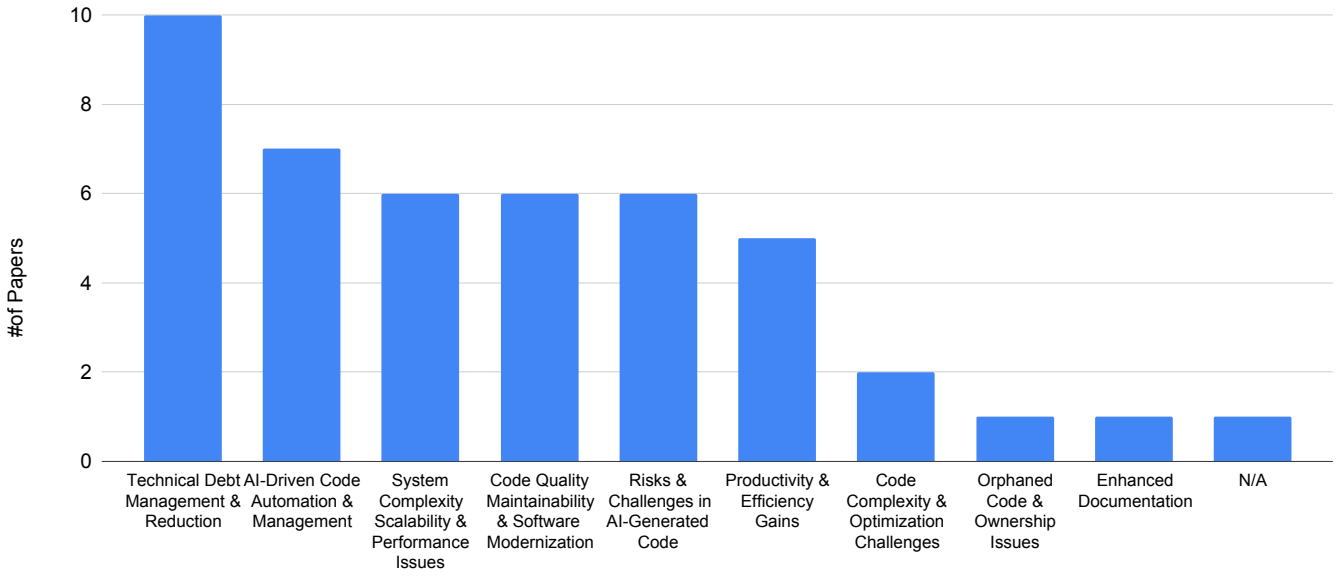

Figure 8: RQ_2: Impact of AI on TDM studied by different PSs

intervention, and enhance software quality having the potential to positively impact software lifecycle management and mitigating the accumulation of new TD. This trend reflects a broader movement toward integrating intelligent systems that support continuous integration, automated refactoring, and efficient TD management. The intersection of AI and software engineering practices suggests a promising direction for improving the sustainability and adaptability of AI-driven solutions.

Figure 9 provides an in-depth analysis of the various challenges associated with TD as identified in the examined PSs. It is important to note that, also in this case, for some studies [PS31], [PS34], [PS32], [PS40], [PS59], it was not possible to determine specific TD-related issues due to a lack of sufficient details or ambiguous descriptions within these works.

The category with the highest mentions for problems is **Maintainability & Software Maintenance** [PS22], [PS30], [PS33], [PS47], [PS49], [PS55], [PS58]. This indicates an increasing emphasis on ensuring that AI and ML systems maintain their resilience, adaptability, and ease of maintenance over time. Given the rapid evolution of ML models and their dependencies, maintaining software quality and performance in the long term has become a critical factor. The complexity of continuously updating models, managing dependencies, and ensuring compatibility with evolving infrastructure contributes significantly to TD accumulation.

The second most frequently identified category is **MLOps & Automation** [PS24], [PS27], [PS29], [PS38], [PS39], [PS51], highlighting, once again, the ongoing struggle to achieve seamless automation in ML-based systems. While MLOps practices have been increasingly adopted to standardize and streamline model deployment, monitoring, and retraining, achieving a high level of automation remains both an objective and a challenge. The difficulties stem from the need to integrate various tools, orchestrate workflows, and manage data pipelines effectively while mitigating TD-related risks. The complexity of ML automation further compounds TD issues, especially in large-scale, production-grade AI systems where continuous delivery and adaptation are critical.

The last factor, Challenges, is depicted in Figure 10. In this case, challenges were not explicitly identified in three studies [PS31], [PS34], [PS59].

The results presented in the figure indicate that the primary challenges revolve around **Code Quality, Maintainability, and the Development Process** [PS22], [PS33], [PS38], [PS40], [PS55], [PS58]. This finding is consistent with previous research in the impact category, underscoring the ongoing commitment to maintaining high-quality code while ensuring long-term maintainability, particularly during the development phase. The emphasis on these aspects reflects the necessity for robust engineering practices to effectively manage the increasing complexity of AI and ML systems while minimizing TD accumulation.

Another set of notable challenges pertains to **AI & ML-specific TD** [PS5], [PS33], [PS39], [PS49], [PS55], highlighting that many issues related to TD in AI remain unresolved. The unique characteristics of AI-driven systems, such as evolving dynamic dependencies, model drift, and the lack of standardized maintenance strategies, contribute to persistent challenges that require further research and practical solutions.

The category of **AI Governance, Explainability, and Compliance** [PS27], [PS29], [PS30], [PS33] is also a significant challenge. This shows that there is a growing demand for transparency and regulatory adherence in AI systems. As machine learning models are increasingly used in important situations, it is important to ensure that they are fair, easy to understand,

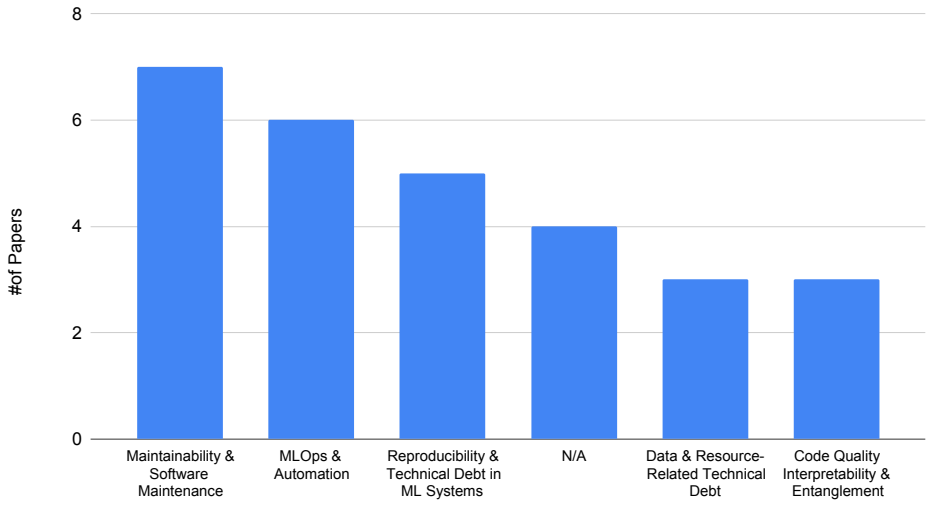

Figure 9: RQ_2: Problems of different PSs

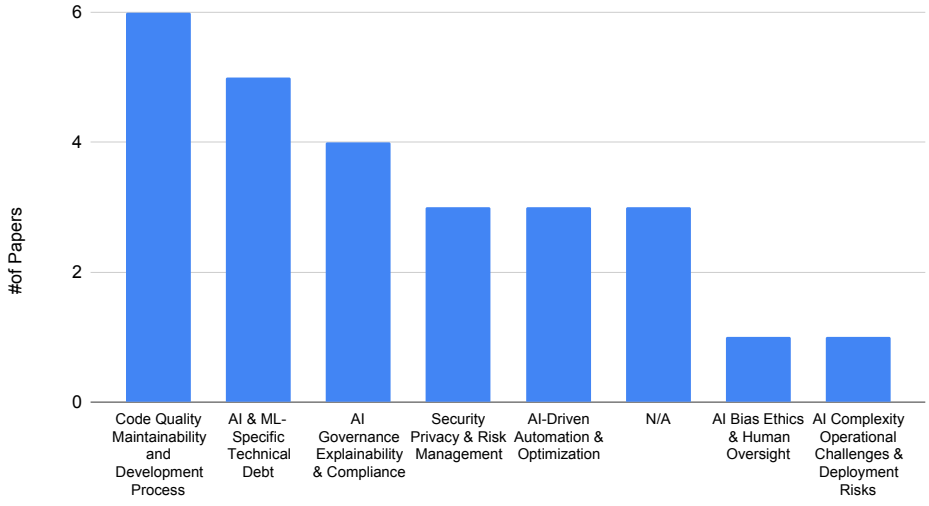

Figure 10: RQ_2: Challenges of different PSs

and follow the law and ethics. The combination of governance and TD shows the need for organized ways to make sure that AI systems are responsible and follow the best practices.

5. Discussion

The results of this research provide a thorough view of how TDM is changing in response to the rise of AI-driven software, and how the integration of AI into software development is impacting the accumulation and management of TD. Specifically, the findings presented in this study are derived from a combined analysis of both white (academic) and grey (practitioner) literature. This dual lens is essential for capturing not only the conceptual evolution of TD in AI-driven systems but also the practical strategies currently employed in the field. Academic sources primarily contribute to the theoretical framing of TD by offering various classifications, taxonomies, and methodological perspectives. On the other hand, grey literature highlights real-world practices, particularly the operationalization of MLOps and the implementation of automation to manage evolving data and model behavior. Recognizing these complementary viewpoints helps ground our discussion in both rigor and relevance. Table 8 summarizes the key differences in findings across these two source types.

5.1. Tailoring TDM for AI-driven software development (RQ_1)

Our analysis indicates that TD specific to ML has emerged as a prevalent concern within the current literature. The unique and dynamic nature of ML, particularly compared to traditional software development practices, demands specialized attention. ML-specific complexities—such as model degradation, evolving performance criteria, and dynamic data de-

Table 8: Summary of Findings by Source Type

Aspect	White Literature (Academic)	Grey Literature (Industry)
RQ1 – TD in ML/GenAI Systems	Conceptualization of ML-specific TD types (e.g., data, model, infrastructure). MLOps introduced as a structured response.	Emphasis on practical MLOps pipelines, tooling issues, and deployment bottlenecks. Frequent references to real-world challenges (e.g., retraining triggers).
RQ2 – Impact of Using AI in SE	Emerging concerns on explainability debt, compliance, and governance. AI-enhanced SE tools discussed conceptually.	Focus on developer productivity, toolchains (e.g., Copilot, ChatGPT), and maintainability risks from GenAI-generated code.
Tooling and Practice	Framework proposals and TD taxonomies.	Hands-on guidance, DevOps/MLOps integrations, and practitioner heuristics.
Perspective	Theory-driven, focusing on design implications and methodological rigor.	Experience-driven, often emphasizing adoption hurdles, informal best practices.

pendencies—pose significant challenges that traditional TD management practices are often ill-equipped to handle effectively.

A key finding from our reviewed studies highlights the critical importance of data-related TDs. Given the central role that data quality and availability play in AI-driven systems, it is clear that robust and integrated data engineering practices are essential. Moreover, the dynamic nature of data itself further amplifies the need for continuous and adaptive MLOps practices. Ensuring sustainable AI systems necessitates proactive management of data debt, underscoring the strategic need for enterprises to integrate comprehensive and dynamically adaptive data governance and engineering frameworks.

MLOps (Machine Learning Operations) is widely recognized as the natural evolution of DevOps for ML-based applications. It extends automation and governance across the full AI lifecycle, with a particular emphasis on data and model management [39]. A key differentiator when compared to DevOps is its emphasis on continuous training, a necessity in real-world deployments where changing data distributions can lead to model degradation. By automating performance monitoring, triggering retraining workflows, and maintaining traceability across data, code, and models, MLOps provides the operational backbone for sustaining data-centric AI systems in production.

Moreover, our findings consistently identified infrastructure and code debt as critical areas requiring attention. Infrastructure TD, closely linked to resource allocation and continuous integration/deployment environments, directly influences operational efficiency and scalability, potentially causing significant performance bottlenecks and increased operational costs if not adequately managed.

A clear and increasingly endorsed solution to these challenges is the adoption of MLOps practices. The growing acknowledgment within both academia and industry of MLOps' potential to systematically mitigate deployment risks associated with ML models underscores its strategic importance. MLOps practices enable enterprises to automate critical activities such as model retraining, ensure reproducibility, and implement robust continuous monitoring mechanisms. This strategic shift toward structured and scalable management practices is vital for sustaining AI-driven systems in competitive production environments.

Furthermore, the critical challenges associated with ML pipelines and feedback loops are precisely the areas where MLOps promises significant improvements. By emphasizing robust automation, continuous monitoring, and resilience, MLOps provides enterprises with proactive tools and methodologies to address and manage TD more effectively.

With the increased adoption of Generative AI tools, implementing an approach specifically tailored for this technology will become essential. Specifically, with the rise of language models, companies are moving towards adopting some of these practices and making use of compact and optimized versions of such models, better known as Short Language Models (SLMs). Looking ahead, similar principles could be equally impactful as enterprises begin adopting SLMs more broadly. Future adaptations of MLOps principles, potentially evolving into frameworks such as Short Language Models Operations (SLMOps), could become essential. This represents a promising avenue for research and innovation, offering strategic advantages to organizations aiming to leverage the full potential of SLMs while maintaining control over TD.

5.2. Impact of AI Use on Technical Debt Management (RQ_2)

The increasing integration of AI technologies into software development environments is not only reshaping how software is built, but also how it evolves over time. A fundamental shift is occurring: software is no longer a static artifact but a continuously evolving entity, shaped by its interaction with data, users, and operating contexts. This evolution—driven by feedback loops, learning mechanisms, and adaptive behaviors—creates a dynamic landscape for TD management that demands new strategies.

Our analysis shows that AI-powered development tools, such as intelligent code generators and automated testing frameworks, offer substantial productivity and maintainability benefits. They streamline development workflows, accelerate time-to-market, and reduce certain types of code-related debt. However, they also introduce novel forms of TD, particularly around system transparency, reproducibility, and maintainability. These tools can generate code or artifacts that are not

easily interpretable, challenging standard documentation and testing practices and potentially leading to long-term maintenance issues.

Additionally, AI introduces a unique form of governance-related TD. As AI systems become more autonomous and embedded in decision-making processes, ensuring compliance with legal, ethical, and operational standards becomes more complex. Organizations must proactively address issues related to explainability, data lineage, and bias to prevent these concerns from compounding into unsustainable technical or reputational debt.

What emerges from our findings is that traditional TD frameworks are no longer sufficient. The fluidity of AI-enhanced systems calls for adaptive and continuous TD management strategies. This reinforces the strategic importance of evolving MLOps into domain-specific operations practices—frameworks that enable continuous integration, monitoring, and mitigation of AI-specific debt forms.

For example, as organizations increasingly deploy generative models and fine-tune them based on usage data, software systems begin to learn and shift autonomously. This self-evolving behavior makes it essential to establish robust traceability, testing, and rollback mechanisms. Companies adopting SLMs are especially positioned to benefit from emerging practices like SLMOps.

From a business perspective, these findings emphasize the urgent need for organizations to develop comprehensive governance strategies, dedicated AI operations workflows, and long-term maintenance plans. Proactively investing in AI-aware TD management practices not only helps mitigate future risks but also positions companies to harness the full transformative potential of AI technologies. Companies that succeed in embedding such operational discipline into their AI initiatives are more likely to maintain software quality, reduce unforeseen maintenance costs, and gain a competitive advantage in rapidly evolving markets.

5.3. Implications for Practitioners

This study provides valuable insights for professionals in software engineering, architecture, and related fields. As AI technologies become increasingly integrated into software systems, TD emerges as both a pragmatic and a strategic concern, affecting system maintainability, development agility, and regulatory compliance.

- First, practitioners should focus on integrating MLOps capabilities into their development pipelines, particularly when working with ML or data-centric systems. This involves setting up tools for continuous integration of models, versioning of datasets, model monitoring, and feedback-informed retraining. Teams should be encouraged to build workflows that are reproducible and observable by design. For instance, a team deploying a fraud detection model could benefit from an MLOps pipeline that automatically triggers retraining based on performance drift, thereby reducing data-related debt.
- Second, teams that are starting to adopt Language Models should plan for TD from the start. This includes auditing and validating AI-generated artifacts, maintaining traceability of prompt engineering and model outputs, and ensuring fallback strategies for failure scenarios. For example, teams deploying GenAI-based assistants can maintain prompt versioning linked to model iterations, helping manage explainability and reproducibility risks.
- Third, TD related to governance—such as explainability, fairness, and data bias—requires close alignment between engineering and compliance units. Practitioners should engage in designing AI systems with built-in controls, documentation standards, and ethical assessment gates. As a practical case, teams working with sensitive user data may automate fairness checks and explainability metrics within their CI/CD workflows to reduce long-term governance debt.
- The advent of SLMOps and AIOps as conceptual extensions of MLOps offers a promising opportunity for practitioners to manage the lifecycle of AI-based systems more efficiently. These frameworks offer a structure for automating processes, establishing accountability, and fortifying resilience across software stacks that incorporate AI. For example, startups adopting SLMs in mobile applications can leverage lightweight monitoring and fine-tuning practices to minimize infrastructure debt while maintaining service quality.

6. Open Issues and Research Gaps

While this review identifies a growing body of work at the intersection of TD and AI-enhanced software development, several important gaps remain. As illustrated in Figure 1, most of the literature clusters around conceptualizing debt in ML and GenAI systems (SE4AI) or highlighting the impact of AI tools on the software lifecycle (AI4SE). However, critical parts of this space are still underexplored. Notably, despite frequent references to MLOps, few studies offer in-depth technical or empirical evaluations of MLOps practices in relation to TD. In many cases, MLOps is either treated as a background assumption or only superficially discussed, without addressing its operational complexity or implementation trade-offs.

In many cases, MLOps is either treated as a background assumption or only superficially discussed, without addressing its operational complexity or implementation trade-offs. Notably, aspects such as security are often treated as marginal or assumed to be inherently managed. However, clearly defining the deployment environment and the supporting toolchain is essential, particularly from a security standpoint. When leveraging managed platforms like AWS, Azure, or Databricks, many security mechanisms—such as network isolation, access control, and compliance—are handled by the provider. In contrast, building a fully open-source MLOps pipeline demands explicit attention to securing each component, as the responsibility for safeguarding data, models, and infrastructure lies

entirely with the development team. This highlights a broader research gap: there is a pressing need for greater attention and dissemination not only around the practical implementation of MLOps, but also around its associated security concerns, which remain largely underexplored in both academic and industrial contexts.

Moreover, while issues such as code and data debt are frequently discussed, other emerging forms (such as governance related debt, explainability debt, or prompt related TD in GenAI) are rarely addressed in a structured manner. In particular, operational frameworks specifically designed for GenAI, such as SLMOps, remain at an early stage, with limited discussion on their practical adoption, standardization, or connection to TDM. Similarly, the rise of prompt engineering introduces new maintenance challenges, including prompt drift, undocumented changes, and strong coupling with downstream outputs, which are currently underexplored in both research and practice. There is also a lack of work on formalizing strategies for managing lightweight generative models and aligning AI system operations with long term compliance and sustainability goals. These open areas represent key opportunities for future research, especially in bridging theory and practice across the SE4AI and AI4SE dimensions.

7. Threats to Validity

In accordance with the established guidelines for secondary studies in software engineering proposed by Ampatzoglou et al. [40], we have structured our discussion of threats to validity into three categories: study selection validity, data validity, and research validity. For each of these, we outline potential threats and describe the mitigation strategies applied. Table 9 summarizes all the identified Threats to Validity (TVs) and their respective Mitigation Approaches (MAs). The TV column corresponds to the ID of each threat to validity in the guidelines [40] and missing IDs suggest that the specific threat is not applicable to our study.

Study Selection Validity: Study selection validity refers to the degree of which relevant PSs are comprehensively and consistently identified and selected. To mitigate the risk associated with incomplete or biased study selection *(TV1)*, we adopted a systematic search protocol. This, thoroughly described in Section 3, includes querying the search string from the most representative sources in the software engineering domain (i.e. *ACM Digital Library, IEEEXplore Digital Library, Web of Science, and Scopus*), ~~applying~~ and the application of a piloted search string, ~~and performing backward snowballing~~. To reduce subjectivity, the studies were selected through a rigorous two-stage screening process (initially by title and abstract, then by full text) with disagreements resolved collaboratively *(TV7)*

To capture insights from current practitioners, grey literature was included in the study. This aligns with our multivocal objective *(TV6)*. We recognize that the inclusion of non-peer-reviewed sources may introduce potential bias. However, we applied a consistent quality assessment strategy to all entries to mitigate this potential challenge. The possibility of bias due to language cannot be discounted. This is due to the fact that only English-language sources were considered *(TV3)*. However, this is typical in software engineering MLRs, due to the dominance of English in technical discourse.

Duplicate publications were handled carefully by retaining only the most informative version *(TV5)*.

Data Validity: Data validity concerns the correctness, consistency, and completeness of the data extracted from primary studies. Therefore, in order to reduce errors in data extraction *(TV13)*, we applied a piloted extraction template and performed cross-checking on a subset of studies. The data was coded according to predefined categories that align with our research questions; the extraction schema was refined iteratively *(TV9)*. Despite our efforts to ensure consistent interpretation, we acknowledge that the qualitative nature of the data may introduce a degree of interpretive subjectivity *(TV14, TV16)*. To overcome this challenge, we triangulated findings across multiple sources and used team discussions to calibrate interpretations.

The classification schema was adapted from prior secondary studies and refined during the coding process to improve fit and clarity *(TV15)*. Although we included both white and gray literature sources to reduce publication bias *(TV10)*, we acknowledge that certain trends, especially in industrial practice, may not have been fully reported.

Research Validity: Research validity addresses the overall credibility, repeatability, and generalizability of the study's design and findings. We ensured repeatability *(TV17)* by making our full replication package publicly available[6]. This includes the search strings, selection protocol, inclusion/exclusion decisions, and coding schema, enabling full transparency and reproducibility of our process. The chosen MLR methodology is well-aligned with our dual research focus on SE4AI and AI4SE *(TV18)*, and the research questions were iteratively validated to maintain alignment with study objectives *(TV19)*.

8. Conclusions

This study offers a comprehensive examination of how TDM practices must evolve to meet the challenges and opportunities introduced by AI-driven software development. Through a multivocal literature review, we identified how the dynamic and data-centric nature of ML and GenAI systems introduces new forms of TD—particularly in areas such as data, infrastructure, and governance—that are not adequately addressed by traditional software engineering practices.

Our findings underscore the strategic importance of MLOps as a response to the complexities of ML-specific TD, particularly in ensuring automation, scalability, and maintainability across the model lifecycle. Moreover, we highlighted how emerging practices, like SLMOps, will support organizations in managing the operational implications of adopting compact and efficient generative models.

The evolution of software from a static artifact to a continuously adapting system demands adaptive TD management approaches. As AI technologies become more embedded in business operations, organizations must adopt forward-looking,

Table 9: Threats to Validity and Mitigation Approaches

TV	Threat to Validity (TV) Description	Mitigation Approach (MA)
TV1	Inadequate identification of PSs	Systematic search across 4 digital libraries (IEEE, ACM, Scopus, Springer) ~~and snowballing~~.
TV3	Language bias	Inclusion limited to English-language PSs; justified by language norms in SE and AI fields.
TV5	Duplicate publications	Retained most recent or complete version when duplicates were detected.
TV6	Grey literature bias	Applied consistent quality assessment; used in combination with academic literature.
TV7	Selection bias due to reviewer subjectivity	Dual screening with conflict resolution through consensus discussion.
TV9	Inadequate extraction schema	Piloted and iteratively refined data extraction form aligned with RQs.
TV10	Publication bias	Inclusion of grey literature to reduce reliance on peer-reviewed venues.
TV13	Data extraction error	Pilot-tested extraction form; subset cross-checked by additional coder.
TV14	Subjectivity in coding	Coder calibration sessions; definitions and coding schema refined collaboratively.
TV15	Inadequate classification schema	Based on prior studies and adapted based on observed themes in data.
TV16	Over-interpretation of qualitative trends	Used triangulation across sources and transparent reasoning.
TV17	Lack of repeatability	Full replication package made publicly available[6].
TV18	Misalignment with review methodology	Followed MLR principles tailored to SE and AI context.
TV19	Poorly scoped/ambiguous RQs	RQs reviewed and refined collaboratively during study design.
TV22	Limited generalizability	Broad inclusion criteria used; time-sensitivity acknowledged due to fast AI evolution.

context-aware strategies that balance innovation with maintainability, and automation with governance.

This review also uncovered substantial implications for practitioners, who must adapt their workflows, governance structures, and toolchains to meet the demands of AI-integrated systems. It further points to a growing convergence of software engineering and operations.

Future research should further investigate how operational frameworks specifically tailored for GenAI such as SLMOps can be matured, standardized, and integrated into organizational workflows. As GenAI practices such as prompt engineering continue to evolve, their potential impact on TD also warrants further exploration. Addressing these needs will be crucial to ensuring that the adoption of AI enhances long-term software sustainability rather than introducing unmanageable forms of debt.

CRediT authorship contribution statement

Sergio Moreschini: Conceptualization, Methodology, Formal Analysis, Investigation, Writing - Original Draft, Visualization.
Elvira Arvanitoy: Formal Analysis, Investigation, Writing - Original Draft.
Elisavet-Persefoni Kanidou: Formal Analysis, Investigation.
Nikolaos Nikolaidis: Formal Analysis, Investigation.
Ruoyu Su: Formal Analysis, Investigation, Writing - Original Draft.
Apostolos Ampatzoglou: Conceptualization, Methodology, Writing - Original Draft.
Alexander Chatzigeorgiou: Conceptualization, Writing - Original Draft.
Valentina Lenarduzzi:.Conceptualization, Methodology, Writing - Original Draft, Supervision

Acknowledgment

The work has been partially founded by the IndustryX and 6GSoft projects (Business Finland).

This work has been partially funded by the Horizon Europe Framework Programme of the European Union under Grant agreement no 101058479.

References

[1] K. Bhandari, K. Kumar, A. L. Sangal, Artificial intelligence in software engineering: Perspectives and challenges, in: 2023 Third International Conference on Secure Cyber Computing and Communication (ICSCCC), pp. 133–137.
[2] T. McDermott, D. DeLaurentis, P. Beling, M. Blackburn, M. Bone, Ai4se and se4ai: A research roadmap, INSIGHT 23 (2020) 47–51.
[3] D. Lo, Trustworthy and synergistic artificial intelligence for software engineering: Vision and roadmaps, in: 2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE), IEEE, 2023, pp. 69–85.
[4] H. Sofian, N. A. M. Yunus, R. Ahmad, Systematic mapping: Artificial intelligence techniques in software engineering, IEEE Access 10 (2022) 51021–51040.
[5] U. K. Durrani, M. Akpinar, M. F. Adak, A. T. Kabakus, M. M. Öztürk, M. Saleh, A decade of progress: A systematic literature review on the integration of ai in software engineering phases and activities (2013–2023), IEEE Access 12 (2024) 171185–171204.
[6] N. Rios, M. G. de Mendonça Neto, R. O. Spínola, A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners, Information and Software Technology 102 (2018) 117–145.
[7] H. J. Junior, G. H. Travassos, Consolidating a common perspective on technical debt and its management through a tertiary study, Information and Software Technology 149 (2022) 106964.
[8] J. A. Ruiz, R. A. Aguilar, J. F. Garcilazo, A. A. Aguileta, A tertiary study on technical debt management over the last lustrum, International Journal of Combinatorial Optimization Problems and Informatics 15 (2024) 181–192.
[9] T. Besker, A. Martini, J. Bosch, Managing architectural technical debt: A unified model and systematic literature review, Journal of Systems and Software 135 (2018).

[10] L. Sousa, L. Rocha, R. Britto, Architectural technical debt - a systematic mapping study, in: XXXVII Brazilian Symposium on Software Engineering (SBES '23), Campo Grande, Brazil.
[11] D. Koulla Moulla, E. Mnkandla, H. Oumarou, T. Fehlmann, Technical debt measurement: An exploratory literature review, in: 33rd International Workshop on Software Measurement and 18th International Conference on Software Process and Product Measurement (IWSM-MENSURA'24), Montréal, Canada.
[12] J. Perera, E. Tempero, Y.-C. Tu, K. Blincoe, A systematic mapping study exploring quantification approaches to code, design, and architecture technical debt, ACM Transactions on Software Engineering and Methodology 33 (2024) 1–44.
[13] Z. Khomyakov, Z. Makhmutov, R. Mirgalimova, A. Sillitti, Automated measurement of technical debt: A systematic literature review, in: 21st International Conference on Enterprise Information Systems (ICEIS 2019), Crete, Greece, pp. 95–106.
[14] P. Klimczyk, L. Madeyski, Technical debt aware estimations in software engineering: A systematic mapping study, e-Informatica Software Engineering Journal 14 (2020) 61–76.
[15] R. Alfayez, W. Alwehaibi, R. Winn, E. Venson, B. Boehm, A systematic literature review of technical debt prioritization, in: IEEE/ACM International Conference on Technical Debt (TechDebt), Seoul, Korea.
[16] V. Lenarduzzi, T. Besker, D. Taibi, A. Martini, F. Arcelli Fontana, A systematic literature review on technical debt prioritization: Strategies, processes, factors, and tools, Journal of Systems and Software 171 (2021).
[17] J. Perera, E. Tempero, Y.-C. Tu, K. Blincoe, Quantifying requirements technical debt: A systematic mapping study and a conceptual model, in: 31st International Requirements Engineering Conference (RE), Hannover, Germany.
[18] R. Melo, V. Fagundes, V. Lenarduzzi, W. Barbosa Santos, Identification and measurement of requirements technical debt in software development: A systematic literature review, Journal of Systems and Software 194 (2022).
[19] J. D. S. da Silva, J. G. Neto, U. Kulesza, G. Freitas, R. Reboucas, R. Coelho, Exploring technical debt tools: A systematic mapping study, in: Enterprise Information Systems (ICEIS 2021), volume 455 of *Lecture Notes in Business Information Processing*, Springer, Cham, 2022.
[20] H. Kleinwaks, A. Batchelor, T. H. Bradley, Technical debt in systems engineering—a systematic literature review, Systems Engineering 26 (2023) 675–687.
[21] W. N. Behutiye, P. Rodríguez, M. Oivo, A. Tosun, Analyzing the concept of technical debt in the context of agile software development: A systematic literature review, Information and Software Technology 82 (2017).
[22] J. Villa, J. O. Ocharán-Hernández, J. C. Pérez-Arriaga, X. Limón, A systematic mapping study on technical debt in microservices, in: 10th International Conference in Software Engineering Research and Innovation (CONISOFT), Ciudad Modelo, San José Chiapa, Mexico.
[23] L. F. Ribeiro, M. A. d. F. Farias, M. Mendonça, R. O. Spínola, Decision criteria for the payment of technical debt in software projects: A systematic mapping study, in: Proceedings of the 18th International Conference on Enterprise Information Systems (ICEIS 2016), Setubal, Portugal.
[24] N. S. R. Alves, T. S. Mendes, M. G. de Mendonça, R. O. Spínola, F. Shull, C. Seaman, Identification and management of technical debt: A systematic mapping study, Information and Software Technology 70 (2016).
[25] C. Fernández-Sánchez, J. Garbajosa, A. Yagüe, J. Perez, Identification and analysis of the elements required to manage technical debt by means of a systematic mapping study, Journal of Systems and Software 124 (2017).
[26] Z. Li, P. Avgeriou, P. Liang, A systematic mapping study on technical debt and its management, Journal of Systems and Software 101 (2015).
[27] A. Ampatzoglou, A. Ampatzoglou, A. Chatzigeorgiou, P. Avgeriou, The financial aspect of managing technical debt: A systematic literature review, Information and Software Technology 64 (2015) 52–73.
[28] B. A. Lunde, R. Colomo-Palacios, Continuous practices and technical debt: a systematic literature review, in: 2020 20th International Conference on Computational Science and Its Applications (ICCSA), Cagliari, Italy, pp. 40–44.
[29] M. E. Nielsen, C. Østergaard Madsen, M. F. Lungu, Technical debt management: A systematic literature review and research agenda for digital government, in: G. e. a. Viale Pereira (Ed.), Electronic Government. EGOV 2020, volume 12219 of *Lecture Notes in Computer Science*, Springer, Cham, 2020.
[30] C. Becker, R. Chitchyan, S. Betz, C. McCord, Trade-off decisions across time in technical debt management: A systematic literature review, in: 2018 IEEE/ACM International Conference on Technical Debt (TechDebt), Gothenburg, Sweden, pp. 85–94.
[31] D. Albuquerque, et al., Managing technical debt using intelligent techniques - a systematic mapping study, IEEE Transactions on Software Engineering 49 (2023) 2202–2220.
[32] J. Bogner, R. Verdecchia, I. Gerostathopoulos, Characterizing technical debt and antipatterns in ai-based systems: A systematic mapping study, in: 2021 IEEE/ACM International Conference on Technical Debt (TechDebt), pp. 64–73.
[33] H. Saeeda, M. O. Ahmad, T. Gustavsson, A multivocal literature review on non-technical debt in software development: An insight into process, social, people, organizational, and culture debt, e-Informatica Software Engineering Journal 18 (2024) 240101.
[34] V. Garousi, M. Felderer, M. V. Mäntylä, Guidelines for including grey literature and conducting multivocal literature reviews in software engineering, Information and Software Technology 106 (2019) 101–121.
[35] B. Kitchenham, S. Charters, Guidelines for performing Systematic Literature Reviews in Software Engineering, 2007.
[36] B. Kitchenham, P. Brereton, A systematic review of systematic review process research in software engineering, Information & Software Technology 55 (2013) 2049–2075.
[37] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: A systematic review, Inf. Softw. Technol. 50 (2008) 833–859.
[38] X. Li, S. Moreschini, F. Pecorelli, D. Taibi, Ossara: abandonment risk assessment for embedded open source components, IEEE Software 39 (2022) 48–53.
[39] S. Moreschini, F. Lomio, D. Hästbacka, D. Taibi, Mlops for evolvable ai intensive software systems, in: 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 1293–1294.
[40] A. Ampatzoglou, S. Bibi, P. Avgeriou, A. Chatzigeorgiou, Guidelines for Managing Threats to Validity of Secondary Studies in Software Engineering, Springer International Publishing, Cham, pp. 415–441.

Appendix A: The Primary Studies (PS_s)

[PS1] M. Alahdab, G. Çalıklı, Empirical analysis of hidden technical debt patterns in machine learning software, in: Product-Focused Software Process Improvement: 20th International Conference, PROFES 2019, Barcelona, Spain, November 27–29, 2019, Proceedings 20, Springer, pp.195–202.

[PS2] E. Breck, S. Cai, E. Nielsen, M. Salib, D. Sculley, The ML test score: A rubric for ML production readiness and technical debt reduction, in: 2017 IEEE International Conference on Big Data (Big Data), IEEE, 2017, pp. 1123–1132.

[PS3] D.K. Chaudhary, S. Srivastava, V. Kumar, A review on hidden debts in machine learning systems, in: 2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT), IEEE, 2018, pp. 619–624.

[PS4] K.-H. Chen, H.-P. Su, W.-C. Chuang, H.-C. Hsiao, W. Tan, Z. Tang, X. Liu, Y. Liang, W.-C. Lo, W. Ji, et al., Apache submarine: A unified machine learning platform made simple, in: Proceedings of the 2nd European Workshop on Machine Learning and Systems, 2022, pp. 101–108.

[PS5] Y. Fu, T. Wang, S. Li, J. Ding, S. Zhou, Z. Jia, W. Li, Y. Jiang, X. Liao, MissConf: LLM-Enhanced Reproduction of Configuration-Triggered Bugs, in: Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings, 2024, pp. 484–495.

[PS6] H. Foidl, M. Felderer, R. Ramler, Data smells: Categories, causes and consequences, and detection of suspicious data in AI-based systems, in: Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI, 2022, pp. 229–239.

[PS7] T.B. Ionescu, S. Schlund, C. Schmidbauer, Epistemic debt: A concept and measure of technical ignorance in smart manufacturing, in: Advances in Human Factors and Systems Interaction: Proceedings of the AHFE 2019 International Conference on Human Factors and Systems Interaction, July 24–28, 2019, Washington DC, USA 10, Springer, 2020, pp. 81–93.

[PS8] A. Menshawy, Z. Nawaz, M. Fahmy, Navigating Challenges and Technical Debt in Large Language Models Deployment, in: Proceedings of the 4th Workshop on Machine Learning and Systems, 2024, pp. 192–199.

[PS9] S. Moreschini, V. Lenarduzzi, L. Coba, Towards a Technical Debt for AI-based Recommender System, in: Proceedings of the 7th ACM/IEEE International Conference on Technical Debt, 2024, pp. 36–39.

[PS10] T. Raffin, T. Reichenstein, J. Werner, A. Kühl, J. Franke, A reference architecture for the operationalization of machine learning models in manufacturing, Procedia CIRP, 115 (2022) 130–135.

[PS11] S. Malakuti, Emerging technical debt in digital twin systems, in: 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), IEEE, 2021, pp. 01–04.

[PS12] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, D. Dennison, Hidden technical debt in machine learning systems, Advances in Neural Information Processing Systems, 28 (2015).

[PS13] R. Sharma, R. Shahbazi, F.H. Fard, Z. Codabux, M. Vidoni, Self-admitted technical debt in R: detection and causes, Automated Software Engineering, 29 (2) (2022) 53.

[PS14] A. Shome, L. Cruz, A. Van Deursen, Data smells in public datasets, in: Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI, 2022, pp. 205–216.

[PS15] R.M. Shukla, J. Cartlidge, Challenges faced by industries and their potential solutions in deploying machine learning applications, in: 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), IEEE, 2022, pp. 0119–0124.

[PS16] Y. Tang, R. Khatchadourian, M. Bagherzadeh, R. Singh, A. Stewart, A. Raja, An empirical study of refactorings and technical debt in machine learning systems, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), IEEE, 2021, pp. 238–250.

[PS17] X. Wang, H. Schuster, R. Borrison, B. Klöpper, Technical Debt Management in Industrial ML-State of Practice and Management Model Proposal, in: 2023 IEEE 21st International Conference on Industrial Informatics (INDIN), IEEE, 2023, pp. 1–9.

[PS18] Z. Codabux, M. Vidoni, F.H. Fard, Technical debt in the peer-review documentation of R packages: A rOpenSci case study, in: 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), IEEE, 2021, pp. 195–206.

[PS19] H. Zhang, L. Cruz, A. Van Deursen, Code smells for machine learning applications, in: Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI, 2022, pp. 217–228.

[PS20] S. Sarkar, A model for technical debt in machine learning systems, Data Science Central, August 30, 2022. Available at: https://www.datasciencecentral.com/a-model-for-technical-debt-in-machine-learning-systems/

[PS21] S. Jain, Adopting MLOps to service ML technical debt, Tata Consultancy Services. Available at: https://www.tcs.com/insights/blogs/mlops-control-service-technical-debt

[PS22] I. Bousquette, AI is writing code now. For companies, that is good and bad, The Wall Street Journal. Available at: https://www.wsj.com/articles/ai-is-writing-code-now-for-companies-that-is-good-and-bad-6f19ecdc

[PS23] A. Bhatia, F. Khomh, B. Adams, A.E. Hassan, An empirical study of self-admitted technical debt in machine learning software, arXiv pre-print arXiv:2311.12019, (2023).

[PS24] R. Hirschfeld, Are LLMs Leading DevOps Into a Tech Debt Trap?, DevOps.com. Available at: https://devops.com/are-llms-leading-devops-into-a-tech-debt-trap

[PS25] A. Baruzzo, M. Gabbrielli, The Pragmatic Programmer for Machine Learning, Chapter 5: Designing and Structuring Pipelines. Available at: https://ppml.dev/design-code.html

[PS26] E. Ordax, Hidden Technical Debt in Generative AI Systems, LinkedIn, March 2024. Available at: https://www.linkedin.com/posts/eordax_genai-llmops-ai-activity-7165246182946922496-7u9V

[PS27] P. Jhamta, Effectively Managing Technical Debt with Generative AI, CrossML. Available at: https://www.crossml.com/effectively-managing-technical-debt-with-generative-ai/

[PS28] N. Eddy, GenAI Early Adopters Face Risk of Technical Debt, Techstrong.ai, April 12, 2024. Available at: https://techstrong.ai/articles/genai-early-adopters-face-risk-of-technical-debt/

[PS29] S. Gupta, The AI balancing act, Freshworks, June 3, 2024. Available at: https://www.freshworks.com/theworks/value/generative-ai-technical-debt-risk-reward/

[PS30] S. Jones, Generative AI: get ready to multiply your Technical Debt, MetaMirror, October 19, 2023. Available at: https://blog.metamirror.io/generative-ai-get-ready-to-multiply-your-technical-debt-fd60ee7d7f4e

[PS31] GFT Technologies SE, GFT AI Impact Beta erases technical debt, speeds up digital transformation, GFT Press Release, November 28, 2023. Available at: https://www.gft.com/be/en/news/press-and-news/2023/press-releases/gft-ai-impact-beta-erases-technical-debt-speeds-up-digital-transformation

[PS32] V. Chugh, Hidden Technical Debts Every AI Practitioner Should be Aware of, KDnuggets, July 7, 2022. Available at: https://www.kdnuggets.com/2022/07/hidden-technical-debts-every-ai-practitioner-aware.html

[PS33] B. Doerrfeld, How CIOs navigate generative AI in the enterprise, CIO, March 20, 2024. Available at: https://www.cio.com/article/1313757/how-cios-navigate-generative-ai-in-the-enterprise.html

[PS34] P. Dureja, How does Gen AI address Technical Debt?, Typo, May 2, 2024. Available at: https://typoapp.io/blog/gen-ai-technical-debt

[PS35] B. Herman, D.D. Leybzon, A. Visnjic, How Observability Uncovers the Effects of ML Technical Debt, Data-centric AI, February 16, 2022. Availableat:https://datacentricai.org/blog/how-observability-uncovers-the-effects-of-ml-technical-debt-2/

[PS36] D. Osipov, How to Avoid the Technical Debt of Machine Learning, Datanami, March 9, 2016. Available at: https://www.datanami.com/2016/03/09/how-to-avoid-the-technical-debt-of-machine-learning/

[PS37] Analytics India Magazine, How to Handle Hidden Technical Debt in a Machine Learning Pipeline, SwissCognitive, March 23, 2020. Available at: https://swisscognitive.ch/2020/03/23/how-to-handle-hidden-technical-debt-in-a-machine-learning-pipeline/

[PS38] Y. Li, M. Soliman, P. Avgeriou, Identifying self-admitted technical debt in issue tracking systems using machine learning, Empirical Software Engineering, 27 (6) (2022) 131.

[PS39] V. Tatan, Intro to MLOps: ML Technical Debt, Towards Data Science, March 2019. Available at: https://towardsdatascience.com/intro-to-mlops-ml-technical-debt-9d3d6107cd95

[PS40] Rinish K N, Conquering Technical Debt: Leveraging Generative AI for Efficient Software Development, ThoughtMinds, July 15, 2024. Available at: https://www.thoughtminds.io/conquering-technical-debt-leveraging-generative-ai-for-efficient-software-development/

[PS41] S. Jain, Leveraging MLOps to Service Technical Debt in Manufacturing, Tata Consultancy Services. Available at: https://www.tcs.com/insights/blogs/mlops-control-service-technical-debt

[PS42] K. Pachauri, Machine Learning: Hidden Technical Debts and Solutions, Towards Data Science, August 25, 2020. Available at: https://towardsdatascience.com/machine-learning-hidden-technical-debts-and-solutions-407724248e44

[PS43] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, Machine learning: The high interest credit card of technical debt, in: SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop), Cambridge, MA, 2014, pp. 112.

[PS44] J. Leung, Managing the Technical Debts of Machine Learning Systems, Towards Data Science, August 2022. Available at: https://towardsdatascience.com/managing-the-technical-debts-of-machine-learning-systems-5b85d420ab9d

[PS45] D. Sklavenitis, D. Kalles, Measuring Technical Debt in AI-Based Competition Platforms, in: Proceedings of the 13th Hellenic Conference on Artificial Intelligence, 2024, pp. 1–10.

[PS46] A. Baruzzo, M. Gabbrielli, The Pragmatic Programmer for Machine Learning, Chapter 5: Designing and Structuring Pipelines. Available at: https://ppml.dev/design-code.html

[PS47] S. Meyen, Navigating DevOps Challenges Amid the AI Revolution, DevOpsCon, July 9, 2024. Available at: https://devopscon.io/blog/navigating-devops-challenges-amid-the-ai-revolution/

[PS48] M.A. Meskarian, Navigating the Minefield: Technical, Ethical, and Governance Debt in AI/ML and Generative AI Models, Medium, Published 1.4 years ago. Available at: https://medium.com/@m.a.meskarian/navigating-the-minefield-technical-ethical-and-governance-debt-in-ai-ml-and-generative-ai-models-fd7fa83861e7

[PS49] M. McAteer, Nitpicking Machine Learning Technical Debt, matthewmcateer.me, May 10, 2020. Available at: https://matthewmcateer.me/blog/machine-learning-technical-debt/

[PS50] L. Lakshmanan, No, You Don't Need MLOps: Keep It Simple, Becoming Human: Artificial Intelligence Magazine, September 18, 2022. Available at: https://becominghuman.ai/no-you-dont-need-mlops-5e1ce9fdaa4b

[PS51] B. Popper, Self-healing code is the future of software development, Stack Overflow, December 28, 2023. Available at: https://stackoverflow.blog/2023/12/28/self-healing-code-is-the-future-of-software-development/

[PS52] Tech & Tales, Streamlining ML Workflows: Tackling Technical Debt with MLflow, Delta Lake, and Databricks, Stackademic, June 28, 2024. Available at: https://blog.stackademic.com/streamlining-ml-workflows-tackling-technical-debt-with-mlflow-delta-lake-and-databricks-2fa47d88300f

[PS53] G. Recupito, F. Pecorelli, G. Catolino, V. Lenarduzzi, D. Taibi, D. Di Nucci, F. Palomba, Technical debt in AI-enabled systems: On the prevalence, severity, impact, and management strategies for code and architecture, Journal of Systems and Software, 216 (2024) 112151.

[PS54] M. Zaver, Technical Debt in Machine Learning, Towards Data Science, December 2015. Available at: https://towardsdatascience.com/technical-debt-in-machine-learning-8b0fae938657

[PS55] S. Sarkar, Technical Debt in Machine Learning System – A Model Driven Perspective, Data Science Central, September 9, 2022. Available at: https://www.datasciencecentral.com/technical-debt-in-machine-learning-system-a-model-driven-perspective/

[PS56] S. Majumder, Technical Debts in ML and MLOps, LinkedIn. Available at: https://www.linkedin.com/pulse/technical-debts-ml-mlops-souptik-majumder

[PS57] K. Sanders, Unraveling AI's Hidden Challenge: Technical Debt – An Interview With Tony Lee, The CTO Club, January 30, 2024. Available at: https://thectoclub.com/news/cto-interview-artifical-intelligence-technical-debt-tony-lee/

[PS58] Serial Ai Publisher, Use AI to Reduce Technical Debt, Koneqt, February 23, 2023. Available at: https://www.koneqt.com/ai-to-reduce-technical-debt/

[PS59] AI Code Generation: Benefits, Tools & Challenges, SonarSource. Available at: https://www.sonarsource.com/learn/ai-code-generation/

[PS60] S. Barocas, What is machine learning debt?, O'Reilly Radar, October 26, 2016. Available at: https://www.oreilly.com/radar/what-is-machine-learning-debt/

[PS61] L. Thorneloe, Why Machine Learning Technical Debt is Especially Bad, Society's Backend, March 12, 2024. Available at: https://societysbackend.com/p/ml-technical-debt

[PS62] B. Doerrfeld, Will the Rise of Generative AI Increase Technical Debt?, DevOps.com, January 25, 2024. Available at: https://devops.com/will-the-rise-of-generative-ai-increase-technical-debt/