On AI Safety and Security Technical Debt in Engineering AI-Enabled Systems

MUHAMMAD TUKUR*, Computer Science, University of Birmingham, Edgbaston, UK; CSE, HBKU, Qatar
HAYATULLAHI B. ADEYEMO, Computing and Informatics, Bournemouth University, UK
TAO CHEN, Computer Science, University of Birmingham, Edgbaston, UK
NOUR ALI, Brunel University London, Uxbridge, UK
ANIS ZARRAD, Computer Science, University of Birmingham, Edgbaston, UK
RICK KAZMAN, University of Hawaii, USA
MARCO AGUS, College of Science and Engineering, HBKU, Qatar
RAMI BAHSOON, Computer Science, University of Birmingham, Edgbaston, UK

Artificial intelligence (AI) systems are increasingly deployed in high-stakes domains such as healthcare, autonomous driving, finance, and education. While these systems offer powerful data-driven and adaptive capabilities, their complexity, rapid evolution, and dependence on dynamic data pipelines introduce new forms of engineering liability collectively referred to as AI Technical Debts (AITDs). AITDs arise from root causes spanning data governance, model implementation, algorithm design, architectural decisions, operational processes, documentation practices, and testing adequacy. Unlike conventional technical debt, many AITDs are latent and propagate across tightly coupled AI pipelines, leading to maintenance challenges, reliability degradation, and heightened safety or security risks. Guided by the principles of AI TRiSM (AI Trust, Risk, and Security Management), this study reinterprets technical debt through the interconnected dimensions of trustworthiness, focusing on AI safety and security technical debts. We conduct a systematic review of 60 primary studies and identify 31 distinct types of AITD, which are organized into a root-cause–oriented taxonomy comprising seven classes. The analysis examines how these debts map to 18 trust-related concerns, including 6 safety hazards and 12 security vulnerabilities. To support mitigation, the review synthesizes 34 actionable guidelines (8 safety and 26 security) targeting the prevention, detection, and reduction of AITDs across the AI lifecycle. Building on these findings, we introduce AITD-MAP, an integrated framework that connects AITD taxonomy, quality and risk impacts, and mitigation strategies into a unified structure for risk-aware AI engineering. The framework aims at assisting AI Software Engineers in making AI Safety and Security technical debts visible, along their root causes and mitigating their presence.

CCS Concepts: • **Security and privacy** → *Systems security*; • **Computing methodologies** → **Artificial intelligence**; • **Software and its engineering** → **Software creation and management**.

Additional Key Words and Phrases: Artificial Intelligence, Mitigation Strategies, Safety, Security, Technical Debt

*Corresponding author

Authors' Contact Information: Muhammad Tukur, mmt310@student.bham.ac.uk, Computer Science, University of Birmingham, Edgbaston, UK; CSE, HBKU, Qatar; Hayatullahi B. Adeyemo, Computing and Informatics, Bournemouth University, Bournemouth, UK, hadeyemo@bournemouth.ac.uk; Tao Chen, Computer Science, University of Birmingham, Edgbaston, Birmingham, , UK, t.chen@bham.ac.uk; Nour Ali, Brunel University London, Uxbridge, London, , UK, Nour.Ali@brunel.ac.uk; Anis Zarrad, Computer Science, University of Birmingham, Edgbaston, Birmingham, , UK, a.zarrad@bham.ac.uk; Rick Kazman, University of Hawaii, Honolulu, , USA, kazman@hawaii.edu; Marco Agus, College of Science and Engineering, HBKU, , , Qatar, magus@hbku.edu.qa; Rami Bahsoon, Computer Science, University of Birmingham, Edgbaston, Birmingham, , UK, r.bahsoon@bham.ac.uk.

Manuscript submitted to ACM

ACM Reference Format:
Muhammad Tukur, Hayatullahi B. Adeyemo, Tao Chen, Nour Ali, Anis Zarrad, Rick Kazman, Marco Agus, and Rami Bahsoon. 2026. On AI Safety and Security Technical Debt in Engineering AI-Enabled Systems. 1, 1 (July 2026), 64 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Artificial Intelligence (AI)-based systems have become deeply embedded in critical domains such as healthcare [57], autonomous vehicles [27], education [198], and finance [77]. An AI-based system is defined as a software-enabled environment that integrates AI techniques—such as machine learning (ML), deep learning (DL), or natural language processing (NLP)—to perform complex tasks with a degree of autonomy or intelligence beyond what is achievable through conventional rule-based automation [164]. These systems leverage such techniques to automate decision-making, adapt to dynamic environments, and solve problems that require reasoning and learning, thereby exceeding the capabilities of traditional rule-based systems [164].

Despite their growing capabilities and adoption, AI-enabled systems introduce substantial engineering and operational challenges. These include issues of transparency, maintainability, robustness, and long-term reliability, especially in dynamic, data-intensive, and safety-critical environments [171]. A significant contributor to these challenges is the accumulation of AI Technical Debt (AITD)—the hidden costs and risks associated with suboptimal architectural, design, or operational decisions made throughout the AI system lifecycle [54]. In this study, AI Technical Debt (AITD) is defined as "a metaphor to indicate the typical quality concerns of suboptimal solutions integrated into the building process of AI-enabled systems" [30]. This definition extends the classical notion of technical debt to the AI context, reflecting quality concerns unique to data-driven and learning-based development processes. As Sculley et al. [178] observed, "ML systems have a special capacity for incurring technical debt, because they have all the maintenance problems of traditional code plus an additional set of ML-specific issues." These issues include data instability, model brittleness, algorithmic bias, and the opacity of learned behaviors. Similar to conventional technical debt, AI Technical Debts (AITDs) may remain latent and only become visible when they manifest as performance degradation, ethical violations, or system failures. However, in AI-enabled systems, this latency is often intensified by uncertainty in the system behaviour, continuous learning, feedback loops, and complex data pipelines and drifts feeding into the learning, making tracing and identifying the root causes of the debts difficult - particularly in high-stakes domains [84, 171].

In response to the growing need for trustworthy AI, the concept of AI TRiSM (AI Trust, Risk, and Security Management) has emerged as a guiding governance framework. Highlighted by Gartner as a top strategic technology trend for 2023 [79] and formally developed by Habbal et al. [85], IBM [95], NIST [147], and Avivah [21], AI TRiSM frames trust as a multidimensional construct encompassing safety, security, fairness, reliability, and transparency. Within this framework, safety and security are treated as complementary and interdependent pillars, not siloed concerns.

AI safety refers to the assurance that AI systems operate reliably without causing unintended harm [17, 84], while AI security focuses on protecting these systems against adversarial threats, unauthorized access, and malicious manipulation [78, 176]. Although their technical origins differ—accidental failures versus intentional attacks—their consequences frequently overlap. A security breach may trigger unsafe system behavior, while weaknesses in safety mechanisms can expose systems to exploitation. In adaptive and continuously learning AI environments, AI Safety and Security vulnerabilities make the system more "porous" (e.g., numerous gaps/holes weakening the protective safety/security boundary of the system) due to factors such as data drift, continuous model updates, feedback loops, and growing system autonomy, which allow failures and attacks to propagate across traditionally separate concerns. This

can be observed through coupled safety–security incidents, shared failure modes, and the accumulation of interrelated technical debts. Despite this convergence, existing literature largely treats safety and security debts in isolation, limiting our ability to reason about their co-evolution and cumulative impact on system risk and trustworthiness over time. The AI TRiSM perspective therefore motivates integrated approaches that jointly identify, model, and mitigate these intertwined forms of debt.

While our study does not extend AI TRiSM in a technical implementation sense, it adopts AI TRiSM as a conceptual lens to reinterpret technical debt in AI-enabled systems. Our core innovation is to view AITDs not solely through architectural layers or lifecycle phases, but also through the risk-oriented dimensions of trust, risk, safety, and security, as advocated by AI TRiSM. This trust-oriented framing enables a structured re-evaluation of technical debt and helps consolidate fragmented discussions around AI safety and security. For example, debts such as ethical debt, undeclared consumers, feedback loops, and algorithmic bias are reinterpreted as systemic risks that degrade AI trustworthiness if left unmanaged. To the best of our knowledge, this is the first systematic review to explicitly examine AITDs through the lens of AI TRiSM. By doing so, we offer a risk-informed, actionable roadmap that aligns technical debt management with the broader goals of trustworthy and sustainable AI governance.

1.1 Motivation

AI-based systems are increasingly deployed in dynamic, high-stakes environments where long-term reliability and perceived trustworthiness are critical [10]. These systems often incur AI Technical Debt (AITD)—hidden costs arising from suboptimal decisions during design, development, or deployment [178]. Unlike traditional technical debt, AITDs are frequently driven by AI-specific factors such as data drift, opaque models, feedback loops, and algorithmic bias, making them harder to detect and manage. In the context of responsible AI, safety and security represent core system quality attributes that can be explicitly designed, implemented, and verified, while trust and risk reflect higher-level assessments of system behavior, assurance, and stakeholder confidence [115, 147]. Recent governance frameworks, including AI TRiSM (AI Trust, Risk, and Security Management), emphasize the need to manage these dimensions in a coordinated manner [21, 79, 85, 95]. However, existing AITD research rarely examines how accumulated technical debt degrades safety and security properties and, in turn, shapes system-level risk exposure and trust outcomes over time.

This study is motivated by the need to bridge this gap. We adopt AI TRiSM as a conceptual lens—not to extend the framework but to reframe AITD through its dimensions. Our goal is to support a more holistic, risk-informed understanding of technical debt in AI systems, enabling more trustworthy and sustainable AI development.

1.2 Research Questions and Contributions

To address the gaps identified in the current literature and advance a root-cause–oriented understanding of AI Technical Debt (AITD), this study conducts a systematic review investigating the forms, impacts, and mitigation strategies associated with AITDs across the AI development lifecycle. The review is guided by the following research questions:

- **RQ1:** What are the different types of technical debt found in AI-enabled systems? And which types are reported most frequently in the literature?
  This question seeks to identify, characterize, and classify AITDs across the full AI lifecycle, including debts related to data, models, algorithms, architecture, operational processes, documentation, and testing.
- **RQ2:** How are the identified AITDs related to safety and security debt?

This question examines the extent to which AITDs contribute to vulnerabilities and operational hazards, reinforcing the AI TRiSM view of trustworthiness.

- **RQ3:** What mitigation strategies and management activities are suggested for addressing the identified AITDs? *This question synthesizes actionable guidelines to support responsible, risk-aware AI development.*

Building on these research questions, the study offers the following key contributions:

- A comprehensive, root-cause–oriented taxonomy of *31 AITDs*, organized into seven major classes: Data & Library–Related Debts, Model & Code–Related Debts, Algorithm–Related Debts, Design & Architecture Debts, Operational & Lifecycle Debts, Documentation & Communication Debts, and Testing & Quality Assurance Debts.
- A *novel mapping* between AITDs and trust-related risks, identifying *6 safety* and *12 security* concerns directly linked to specific forms of debt, thereby demonstrating how AITDs shape vulnerability paths and operational hazards in AI-enabled systems.
- A synthesis of *34 actionable mitigation guidelines* (*8 safety* and *26 security*), providing concrete strategies for preventing, detecting, and reducing AITDs. These include practices such as Human-AI Control Mode Switching, Ethical Black Boxes, Continuous Behavioral and Drift Monitoring, Adversarial and Defensive Training, Out-of-Distribution Detection, and Formal Verification for high-risk applications.
- The introduction of AITD-MAP, a unified analytical framework that integrates AITD taxonomy, impact analysis, and mitigation strategies (See Figure 1), designed to support risk-informed decision-making in AI system engineering and align with principles of trustworthy AI.

By consolidating fragmented insights and providing a structured, risk-aware roadmap for managing AI Technical Debt, this study advances responsible, resilient, and sustainable AI engineering. The findings reinforce the importance of integrating quality, safety and security considerations throughout the AI lifecycle, offering actionable guidance for practitioners and researchers working toward trustworthy and dependable AI systems.

1.3 Paper Organization

The rest of the paper is structured as follows: Section 2 presents the related works. Section 3 describes the research methodology. Section 4 presents the taxonomy of AITDs. It further explores their impact on software quality attributes. Section 5 examines their connection to safety and security. Section 6 proposes mitigation guidelines. Section 7 discusses principal findings and outlines strengths and limitations. Section 8 concludes with a summary and future directions.

2 Related work

Technical debt (TD) has been extensively studied in the context of traditional software [20] and systems [109, 110] engineering. Foundational reviews have explored its identification [14], management [20, 119], prioritization [11], financial implications [15], and intelligent management techniques across diverse domains [9]. However, these studies generally overlook the unique characteristics and quality concerns associated with AI-based systems.

In response to the growing importance of Artificial Intelligence (AI), recent research has begun to explore technical debt in AI-enabled systems. These efforts, however, remain fragmented. Some works focus on specific debt types such as ethical debt [158], bias and fairness [168], self-admitted technical debt (SATD) [148], algorithmic debt [190], architectural debt [175], and requirements engineering debt [26]. Others investigate technical debt within particular application domains, such as AI-based competition platforms [193], recommender systems [138], large language models (LLMs) [135], and complex systems [26].

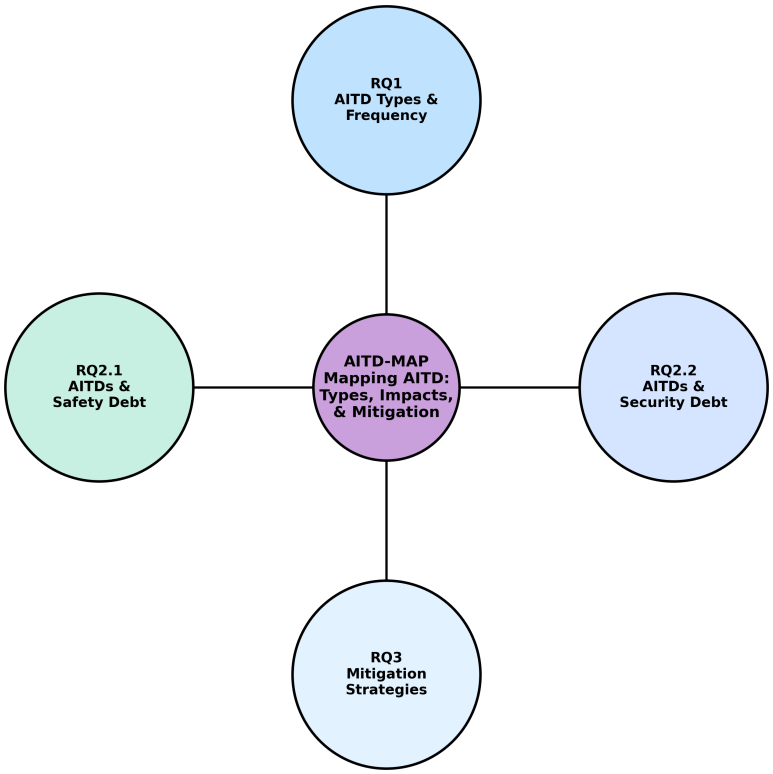

Fig. 1. **Circular representation of AITD-MAP (Mapping AI Technical Debt: Types, Impacts, & Guidelines)**, illustrating how each research question (RQ1–RQ3) contributes to the construction of the integrative framework. The diagram highlights the flow from taxonomy development and frequency analysis to the assessment of security and safety concerns and corresponding mitigation strategies.

In terms of comprehensive analyses, Bogner et al. [30] presented a systematic mapping study on TD and anti-patterns in AI systems, offering a partial taxonomy with limited impact analysis and scattered mitigation approaches. Recupito et al. [164] conducted a survey-based study that captures practitioner insights into architecture-level AITDs, but lacks a structured taxonomy and generalized mitigation framework. Washizaki et al. [215] attempted to classify SE design patterns for ML systems but focused more on conceptual classification than technical debt characterization.

Sculley et al. [178] provided the foundational industry perspective on ML-specific technical debt, such as glue code and entanglement, but did not offer empirical classification or mitigation strategies. Bhatia et al. [28] conducted a large-scale empirical study on SATD in ML projects, contributing a refined SATD taxonomy and insights into long-term debt evolution, although security and safety aspects were not addressed. Menshawy et al. [135] offered practical insights into TD associated with LLM deployment, highlighting engineering trade-offs and mitigation practices from real-world systems.

A comparative summary of these works is presented in Table 1. As shown, existing literature generally lacks a holistic view of AITDs across types, impacts, and mitigation strategies. In addition, few studies incorporate a systematic approach to evaluate the security and safety implications of AITDs or provide actionable guidelines for debt reduction.

In contrast, this study provides the first comprehensive review that systematically identifies and categorizes 31 distinct types of AITDs from 60 primary studies. It goes further to analyze their effects on system quality, with an emphasis on safety and security concerns, and synthesizes 16 actionable guidelines for mitigating these issues. Our findings address critical gaps in the previous literature by integrating taxonomic classification, impact assessment, and practical mitigation strategies within a single unified framework.

Table 1. Summary of Related Works on Technical Debt in AI-Based Systems

Study	Type	# of Studies	Pub. Period	Objective	Methodological Framework	Taxonomy Provided	Impact Analysis	Security/ Safety Consideration	Mitigation Strategies
Bogner et al. (2021) [30]	Systematic Mapping Study (SMS)	21	Up to 2020	Map the landscape of TD and antipatterns in AI systems	Wohlin [217] and Petersen [157] guidelines	Partial – categorized TDs & antipatterns loosely	General discussion of impact	Mentioned generally	46 scattered mitigation approaches
Recupito et al. (2024) [164]	Survey Study	53 practitioner responses	Up to 2024	Investigate code- and architecture-level AITDs	Survey + qualitative content analysis	Focused on 9 AITD types	Moderate, practitioner-perceived impact	Mentioned generally	Mostly ad hoc, manual strategies
Sklavenitis et al. (2024) [193]	Scoping Review	100	2012–2023	Measure AITDs in competitions; introduce Accessibility Debt	Scoping review + domain-specific questionnaire	Yes – 18 AITDs incl. Accessibility Debt	Structured analysis with practical examples	Not a primary emphasis	Not a primary emphasis
Washizaki et al. (2019) [215]	Preliminary SLR	38	Up to 2019	Identify SE design patterns in ML systems	Literature review + ML lifecycle classification	Yes – 33 SE patterns	Partially – conceptual	Not explicitly analyzed	Not explicitly analyzed
Sculley et al. (2015) [178]	Conceptual Essay	N/A	Pre-2015	Highlight ML-specific TD from industry view	Conceptual analysis (Google experience)	Yes – anti-patterns	Qualitative discussion	Limited (e.g., feedback loops)	Conceptual suggestions (e.g., monitoring)
Bhatia et al. (2023) [28]	Empirical Study	318 ML + 318 Non-ML	Up to 2023	Analyze SATD in ML software	Mixed methods – SHAP, survival, manual coding	Yes – Extends SATD taxonomy	Evolution + predictors of SATD	Not directly addressed	Implicit suggestions
Menshawy et al. (2024) [135]	Perspective Study	N/A	Up to 2024	Discuss TD in LLM deployment	Experience-based synthesis + examples	Yes – LLM-specific TD types	Covers performance, memory, latency	Explicit (bias, hallucinations, feedback)	Prompt tuning, quantization, caching
Ours (2025)	Scoping Review	60	Up to 2025	Identify, categorize, and analyze AITDs, their impacts, and mitigation strategies	PRISMA-ScR [208]	Comprehensive taxonomy of AITDs	Detailed impact on AI-system trustworthiness	Explicit analysis of security and safety implications	34 synthesized mitigation guidelines

3 Methodology

To address the research questions outlined in the introduction, we adopted the guidelines set by the PRISMA Extension for Scoping Reviews (PRISMA-ScR) [208]. As illustrated in Figure 2.A, this framework provides a structured and systematic approach for conducting comprehensive scoping reviews, ensuring rigor and thoroughness. The literature search was conducted through the following stages:

3.1 Search Strategy

The search strategy encompassed the selection of bibliographic databases, formulation of search terms and strings, establishment of inclusion and exclusion criteria, and the process for selecting relevant studies for inclusion.

3.2 Data Source Selection

To ensure comprehensive coverage of relevant literature, we conducted searches in the following electronic databases: ACM Digital Library, IEEE Xplore, Scopus, and Springer. These databases were selected for their extensive coverage of research in software engineering and computer science in general.

3.3 Venue Selection Criteria

In addition to database and content filtering, venue selection was guided by stringent quality and relevance criteria. Priority was given to studies published in high-impact, peer-reviewed journals, top-tier conferences, workshops, and symposium with a clear focus on AI system engineering and/or software development practices. The final selection spans 34 distinct venues, with CAIN, Empirical Software Engineering, SEAA, ICSE, and TechDebt, among the most represented (c.f. Table 2). Emphasis was placed on venues indexed by ACM, IEEE, Springer, and Scopus, given their established reputation and consistent record of publishing research on software engineering.

3.4 Search terms

AI-enabled systems share certain forms of technical debt with traditional software systems; however, they also introduce unique types of technical debt that arise from their data-driven architectures, learning components, adaptive behaviors and feedback loops, and emergence. Consequently, it is essential to focus on literature that explicitly addresses technical debt in AI-based systems. Based on this distinction, the following terms were identified as most relevant for constructing the search queries.

- AI Terms: Artificial Intelligence, AI, Machine Learning, ML, Deep Learning, DL, Natural Language Processing, NLP, Generative Artificial Intelligence, GenAI, Large Language Model, LLM, Agentic AI.
- Technical Debt Terms: Technical Debt, TD, AITD, Anti-patterns, Self-Admitted Technical Debt, SATD, Smell.

3.5 Search Strings

Search queries were formulated using appropriate literal and semantic synonyms to capture a wide range of relevant results. Table 3 presents the search strings applied to each data source:

3.6 Search Criteria

Studies were included based on their direct relevance to AI-technical debt. The inclusion and exclusion criteria were as follows:

3.6.1 Inclusion criteria:

- **IC1.** The study must explicitly discuss or analyze technical debt within AI-enabled systems and provide direct insights relevant to one or more of the research questions formulated in this review.
- **IC2.** The publication must be a peer-reviewed journal article, conference paper, workshop, or symposium contribution.

Table 2. Distribution of Selected Studies by Venue

S/N	Venue	Count
1	International Conference on AI Engineering: Software Engineering for AI (CAIN)	6
2	Empirical Software Engineering	5
3	Euromicro Conference on Software Engineering and Advanced Applications (SEAA)	4
4	International Conference on Software Engineering (ICSE)	4
5	ACM/IEEE International Conference on Technical Debt (TechDebt)	4
6	ACM Transactions on Software Engineering and Methodology	3
7	IEEE International Conference on Software Maintenance and Evolution (ICSME)	3
8	IEEE Transactions on Software Engineering	2
9	Software Quality: Future Perspectives on Software Engineering Quality (SWQD)	2
10	AI and Ethics	2
11	International Conference on Mining Software Repositories (MSR)	2
12	Journal of Systems and Software	1
13	ACM Conference on Equity and Access in Algorithms, Mechanisms, and Optimization	1
14	ACM Conference on Fairness, Accountability, and Transparency	1
15	ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering	1
16	ACM SIGMOD Record	1
17	ACM/IEEE Workshop on AI Engineering-Software Engineering for AI (WAIN)	1
18	Advances in Neural Information Processing Systems	1
19	IEEE International Conference on Big Data (Big Data)	1
20	Information and Software Technology	1
21	International Conference on Automated Software Engineering (ASE)	1
22	International Conference on Green Computing and Internet of Things (ICGCIoT)	1
23	International Conference on Industrial Informatics (INDIN)	1
24	International Conference on Product-Focused Software Process Improvement	1
25	International Conference on Software Architecture Companion (ICSA-C)	1
26	International Requirements Engineering Conference Workshops	1
27	International Workshop on Empirical Software Engineering in Practice (IWESEP)	1
28	Workshop on Machine Learning and Systems	1
29	World Wide Web Conference	1
30	International Conference on Emerging Technologies and Computing (ICETC)	1
31	Brazilian Symposium on Software Components, Architectures, and Reuse	1
32	International Conference on Program Comprehension (ICPC)	1
33	IEEE Access	1
34	IEEE Annual Computing and Communication Workshop and Conference (CCWC)	1
Total		**60**

- **IC3.** The publication must have been released between 2015 and November 2025. This period begins with the introduction of the concept of technical debt in AI and machine learning systems by Sculley et al. [178] in 2015.
- **IC4.** The paper must exceed three pages in length to ensure adequate depth of discussion and analysis.

3.6.2 Exclusion criteria:

- **EC1.** Non-original research articles were excluded. This includes review papers, Ph.D. dissertations, secondary studies (e.g., SLRs and SMSs), posters, editorials, and magazine articles.
- **EC2.** Articles not written in English were excluded.
- **EC3.** Studies without accessible full-text versions were excluded.
- **EC4.** Studies that do not explicitly address technical debt in AI-based systems were excluded.

Table 3. Restricted Electronic Search Results for AI and Technical Debt

Data Source	Search String	Filters Applied	Results
ACM Digital Library	[[Abstract: artificial intelligence] OR [Abstract: ai] OR [Abstract: machine learning] OR [Abstract: ml] OR [Abstract: deep learning] OR [Abstract: dl] OR [Abstract: natural language processing] OR [Abstract: nlp] OR [Abstract: generative artificial intelligence] OR [Abstract: genai] OR [Abstract: large language model] OR [Abstract: llm] OR [Abstract: agentic ai]] AND [[Abstract: technical debt] OR [Abstract: td] OR [Abstract: artificial intelligence technical debt] OR [Abstract: aitd] OR [Abstract: anti-patterns] OR [Abstract: self-admitted technical debt] OR [Abstract: satd] OR [Abstract: smell]]	Research articles only, and date	330
IEEE Xplore	(("Abstract":"Artificial Intelligence" OR "Abstract":"AI" OR "Abstract":"Machine Learning" OR "Abstract":"ML" OR "Abstract":"Deep Learning" OR "Abstract":"DL" OR "Abstract":"Natural Language Processing" OR "Abstract":"NLP" OR "Abstract":"Generative Artificial Intelligence" OR "Abstract":"GenAI" OR "Abstract":"Large Language Model" OR "Abstract":"LLM" OR "Abstract":"Agentic AI") AND ("Abstract":"Technical Debt" OR "Abstract":"TD" OR "Abstract":"Artificial Intelligence Technical Debt" OR "Abstract":"AITD" OR "Abstract":"Anti-patterns" OR "Abstract":"Self-Admitted Technical Debt" OR "Abstract":"SATD" OR "Abstract":"Smell"))	Journals and conferences only, and date	558
Scopus	("Artificial Intelligence" OR "AI" OR "Machine Learning" OR "ML" OR "Deep Learning" OR "DL" OR "Natural Language Processing" OR "NLP" OR "Generative Artificial Intelligence" OR "GenAI" OR "Large Language Model" OR "LLM" OR "Agentic AI") AND ("Technical Debt" OR "TD" OR "Artificial Intelligence Technical Debt" OR "AITD" OR "Anti-patterns" OR "Self-Admitted Technical Debt" OR "SATD" OR "Smell")	Document Type: Article and Conference paper; Subject area: Computer Science and Engineering; Language: English only; and date	203
Springer	"Artificial Intelligence" OR "Machine Learning" OR "Deep Learning" AND "Technical Debt" OR "Anti-patterns"	Research articles and conference papers only; Discipline: Computer Science; Subject area: Software Engineering, Software Testing; Language: English only; and date	260
Total			**1,351**

- **EC5.** Duplicate publications were excluded. These include instances where the same study appeared in multiple databases or was published in more than one venue (e.g., journal, conference, or workshop).

3.7 Study Selection

The study selection process was conducted in three phases: (1) automatic search restriction and duplicate removal, followed by (2) initial screening of titles and abstracts for relevance, and (2) a full-text review to confirm eligibility based on the predefined inclusion and exclusion criteria.

3.7.1 Automatic search restriction and duplicate removal. In this phase, we included papers published between January 2015 and October 2025, as Sculley et al. [178] were the first to introduce the concept of technical debt in AI and machine learning systems in the year 2015. In addition to the time restriction, we applied additional filters based on the options available in each digital library. For example, the Scopus Digital Library allows filtering by document type, subject

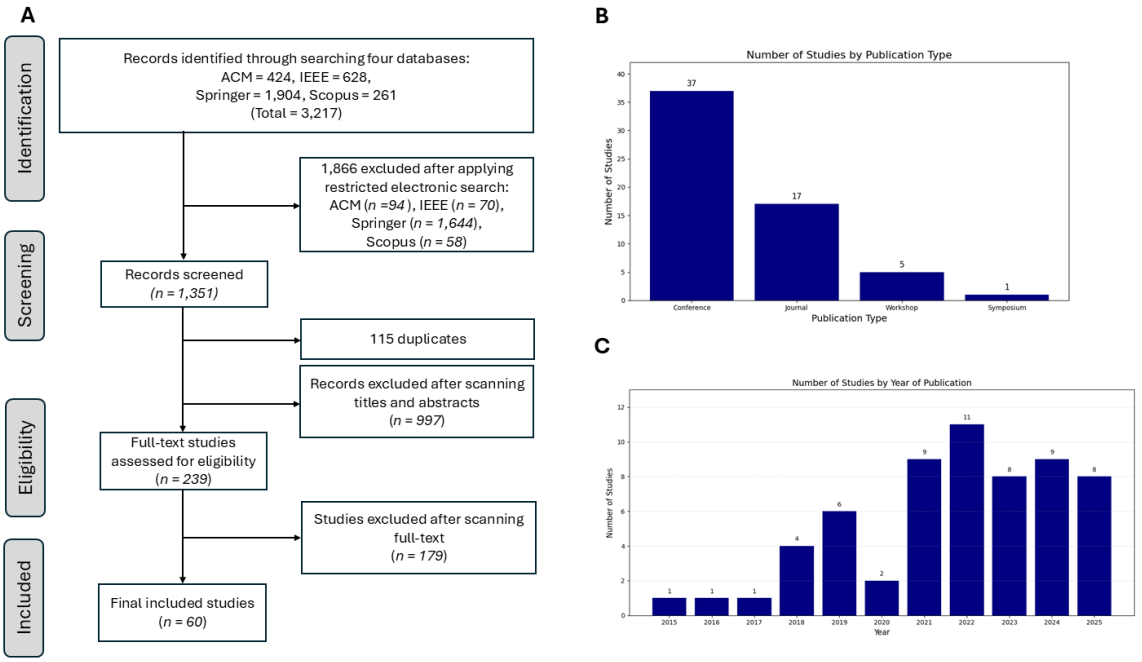

Fig. 2. Methodology charts: (A) PRISMA chart of the included studies; (B) publication type of the selected papers. (C) The distribution of studies over the years.

area, language, and publication date. Table 3 outlines the filters applied and the number of studies retrieved from each digital database. In total, 1,351 studies were initially retrieved, and after removing 115 duplicates, 1,236 unique papers with distinct titles and abstracts were considered for further analysis. Further information and replication package is provided in the *Search Results* folder of the Supplementary Material for additional details.

3.7.2 Screening based on title and abstract. In this phase, studies were screened based on their titles, abstracts, and the availability of full texts. This resulted in the exclusion of 997 studies due to irrelevance or inaccessible full-texts, leaving 239 unique full-text studies for further evaluation.

3.7.3 Screening based on full-text. The remaining studies were subjected to a full-text review, where the inclusion and exclusion criteria outlined in Sec. 3.6 were rigorously applied. Studies that only discussed technical debt without specifically addressing AI-based systems, or those that did not link technical debt to AI-enabled systems, were excluded. Following a comprehensive review, 60 studies were identified as highly relevant and selected as primary sources. The multidisciplinary nature of this review is evident in the diverse range of publication venues, as outlined in the references (c.f. 2. Figure 2.B illustrates the types of publications included, while Figure 2.C shows the distribution of the selected studies over the past decade.

3.8 Grounded Theory Coding Procedure

To systematically identify and categorize the 31 AI Technical Debts (AITDs), we adopted a grounded theory methodology structured around the three classical phases of Corbin and Strauss [49]: *open coding*, *axial coding*, and *selective*

coding. During the open coding phase, two co-authors independently analyzed the full-text content of the 60 primary studies, extracting and labeling recurring technical challenges indicative of debt-like behavior. This stage produced 101 initial codes, with each refined AITD supported by at least three underlying conceptual indicators. In the axial coding phase, these codes were iteratively compared, clustered, and refined to identify conceptual relationships and remove redundancies. Through this constant comparison process, the analysis converged into 31 distinct AITDs, each representing a higher-level abstraction of related debt patterns. In the selective coding phase, these 31 AITDs were organized into seven main root-cause-oriented categories. This classification underwent a structured assessment involving the co-authors. Agreement was achieved through multiple consensus-building sessions, with disagreements resolved via structured deliberation until full consensus was reached. The analysis achieved theoretical saturation after three iterations, at which point no new debt concepts emerged. This rigorous multi-stage approach ensured the reliability, transparency, and saturation of the resulting taxonomy, as summarized in Tables 4 and further detailed in Section 4. Additional details are provided in the Supplementary Material.

4 Overview of the identified AITDs and their taxonomy (RQ1)

This section offers a comprehensive definition and discussion of the thirty-one (31) AI Technical Debts (AITDs) identified in the primary studies, each illustrated with relevant use cases. The AITDs are systematically categorized, and a detailed analysis is conducted, with each debt ordered by frequency of occurrence using grounded theory as the analytical approach [49] (see Tables 4 and the Supplementary Material). Additionally, an in-depth examination of the overall impact of AITDs on the quality of AI-based systems is provided.

4.1 Root-Cause-Oriented AITD Taxonomy

To develop a unified and conceptually coherent understanding of the diverse forms of AI Technical Debt (AITD), the thirty-two identified debts were systematically classified according to their root causes—that is, the underlying technical, organizational, or ethical conditions that give rise to debt accumulation within AI-enabled systems. This root-cause-oriented perspective emphasizes the intrinsic source of each debt rather than its surface manifestation, providing a more holistic view of how AITDs emerge, propagate, and persist throughout the lifecycle of intelligent systems. The resulting taxonomy consolidates related technical debts into seven principal categories, each representing a distinct causal domain that influences the introduction and long-term impact of debt:

(1) Data & Library–Related Debts
(2) Model & Code–Related Debts (Implementation Debts)
(3) Algorithm–Related Debts
(4) Design & Architecture Debts (Structural Debts)
(5) Operational & Lifecycle Debts
(6) Documentation & Communication Debts
(7) Testing & Quality Assurance Debts

Each category groups together debt types that share a common origin, engineering failure mode, or decision-making trade-off, enabling more precise reasoning about how and why these debts arise. This structure also facilitates targeted mitigation by linking each debt to its underlying technical or managerial driver. The classification process followed an iterative grounded-theory procedure, in which the co-authors independently reviewed and validated the conceptual boundaries of each identified debt type. Disagreements were resolved through structured, consensus-based discussions,

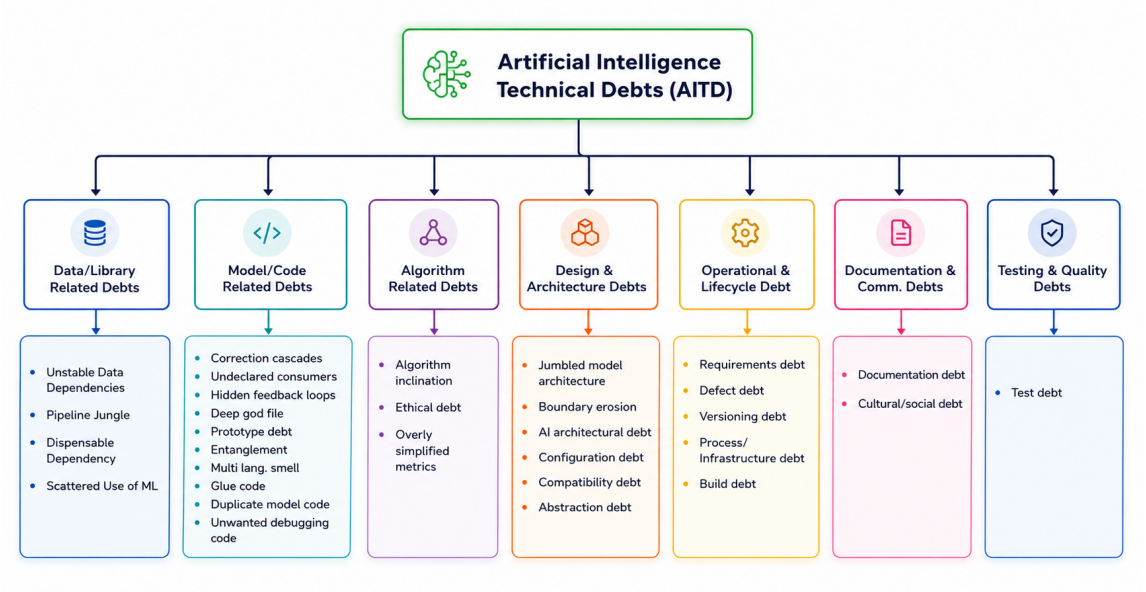

Fig. 3. AITD taxonomy showing seven main categories—each divided into subcategories with corresponding debt types.

resulting in a stable and mutually agreed-upon taxonomy. The hierarchical relationships among the seven categories, their constituent subcategories, and the individual technical debt types are illustrated in Figure 3. This taxonomy provides an integrated and extensible representation of the AITD landscape, synthesizing perspectives from software engineering, machine learning systems, responsible AI, and operational practices. The resulting categorizations are presented in the subsections that follow, each accompanied by detailed descriptions, example scenarios, and a justification for its assignment to the respective category.

4.1.1 Data & Library-Related Debts. These debts arise from shortcomings in data governance, feature-pipeline stability, and dependency management—factors that form the foundation of every AI-based system. Because AI models rely heavily on the quality, consistency, and traceability of their training and inference inputs, even minor irregularities in data or library configurations can propagate significant degradation in system reliability, reproducibility, and long-term maintainability. These debts collectively reflect failures to manage the data ecosystem and software dependencies that support the learning pipeline.

i. **Data Debt** This emerges when datasets used for training, validation, or inference suffer from quality issues such as missing values, biased distributions, poor metadata, untracked schema evolution, or inconsistent preprocessing procedures. In AI systems, where model performance is intrinsically tied to input quality, such deficiencies can severely distort learned representations and weaken generalization. Data Debt often accumulates gradually as datasets evolve without proper versioning, monitoring, or documentation. *Impact:* It introduces instability, model drift, fairness violations, reduced robustness, and increased retraining overhead. Production models may fail unpredictably, degrade silently, or exhibit harmful behavior—such as biased decision-making—when new data deviates from historical patterns. *Use Case:* In a real-time fraud-detection system, a financial institution updates its transaction schema by renaming fields and adding new categorical indicators. Because these changes are

untracked, the deployed model interprets inputs incorrectly, resulting in a surge of false positives and customer complaints. *Justification (Why):* Data Debt is classified under Data & Library-Related Debts because its root cause is inherently tied to data governance failures, unstable data pipelines, and unmanaged dataset evolution—issues central to the foundational data layer of AI systems.
Source(s): [3, 7, 19, 26, 32, 52, 53, 60, 69, 70, 94, 105, 114, 137, 138, 142, 159, 165, 168, 175, 178, 186, 188, 204, 213, 220, 224].

ii. **Pipeline Jungle:** This refers to preprocessing or feature-engineering pipelines that have grown organically—often through rapid experimentation—resulting in deeply nested, ad-hoc, and poorly documented chains of transformations. As components evolve, the pipeline becomes opaque, fragile, and difficult to reproduce or debug. AI systems are particularly vulnerable since the learning process depends critically on deterministic and transparent data transformations. *Impact:* It undermines reproducibility, slows development, and increases the risk of hidden data leaks or inconsistent training-inference paths. Even small upstream changes can cause cascading failures or silent performance degradation. *Use Case:* A computer-vision team incrementally adds new image augmentations for training. Over time, the pipeline incorporates dozens of conditional operations spread across multiple scripts. During deployment, a mismatch between training and inference transformations leads to lower detection accuracy in real-world images. *Justification (Why):* Pipeline Jungle is grouped under this category because its core deficiencies relate to data pipeline management, transparency, and structural oversight—problems rooted in data engineering rather than model architecture or code structure.
Source(s): [7, 26, 70, 138, 164, 178, 186, 213, 215].

iii. **Scattered Use of ML Libraries (SML):** This debt arises when an AI system relies on multiple machine learning libraries, framework versions, or overlapping APIs without clear standardization. Examples include mixing TensorFlow 1.x and 2.x, combining incompatible PyTorch modules, or using different versions of NumPy for different components. Such inconsistency introduces brittleness, dependency conflicts, and unpredictable behavior. *Impact:* SML leads to nondeterministic outputs, runtime incompatibilities, serialization failures, deployment instability, and increased maintenance cost—especially when model components must interoperate in production. *Use Case:* A research team trains a model in PyTorch but uses TensorFlow scripts for feature extraction. Minor version differences between development and production environments cause silent discrepancies in tensor shapes, breaking the deployment pipeline. *Justification (Why):* SML is rooted entirely in dependency mismanagement and inconsistency within the library ecosystem—precisely the type of foundational support issue that characterizes Data & Library-Related Debts.
Source(s): [51, 164].

iv. **Dispensible Dependency:** It refers to unused, obsolete, or redundant software packages and libraries that remain in the environment even though they are no longer required by the AI system. These dependencies increase the risk of security vulnerabilities, version conflicts, and inflated environments that are difficult to reproduce or audit. *Impact:* This debt increases attack surfaces, slows build times, complicates dependency resolution, and introduces uncertainty into model reproducibility. It may also hinder deployment on resource-constrained platforms. *Use Case:* A legacy feature-extraction toolkit remains installed in a production environment despite being replaced months earlier. A vulnerability scanner later identifies a critical security flaw in the unused library, forcing emergency remediation. *Justification (Why):* Dispensable Dependency is placed under Data & Library-Related Debts because its origin lies in library ecosystem mismanagement and poor dependency hygiene—both fundamental aspects of the AI data-processing environment.

Source(s): [32, 42, 51].

All the debts in this category share a common root cause: poor control, monitoring, and governance of the data pipelines and software dependencies that constitute the backbone of AI systems. These debts hinder reproducibility, undermine trust in model behavior, and increase operational friction across the entire AI lifecycle.

4.1.2 Model/Code Related Debts. These originate from deficiencies in implementation practices, modularization, interface management, and the structural integrity of the code that supports AI components. AI systems are particularly susceptible to these debts because model behavior is deeply intertwined with the structure of the underlying code artifacts—such as dataflow logic, model wrappers, feature utilities, and runtime inference pathways. Unlike traditional software components, AI models introduce additional complexity stemming from non-deterministic behaviors, hidden state, and tight coupling between learned parameters and computational code. The debts in this category collectively degrade maintainability, reliability, and integration stability across the model development and deployment lifecycle.

i. **Correction Cascades (CC):** Occur when modifications made to fix a model issue inadvertently introduce new defects elsewhere in the system. In AI pipelines, where components depend heavily on learned representations or preprocessing logic, even minor adjustments—such as threshold tuning, feature normalization changes, or loss-function updates—may ripple through downstream modules. *Impact:* This debt increases maintenance effort, leads to unpredictable regressions, and complicates quality assurance. Over time, the system becomes brittle, with every fix posing a risk of degrading other functionalities. *Use Case:* A development team adjusts the classification threshold of a churn-prediction model to reduce false negatives. The change unexpectedly affects marketing automation logic that relies on threshold-based customer segmentation, resulting in misaligned promotional campaigns. *Justification (Why):* Correction Cascades are categorized as Model & Code–Related Debts because their root cause lies in tight coupling, insufficient isolation, and ad-hoc code modifications that propagate unintended side effects across model components.
Source(s): [26, 164, 178, 189].

ii. **Undeclared Consumers:** This arises when model outputs are consumed by downstream services or processes unknown to the development team. These hidden dependencies often emerge in fast-paced AI development environments where APIs evolve quickly and internal teams reuse AI components without formal registration or interface governance. *Impact:* They create operational risk, unauthorized data exposure, brittle integration pathways, and unpredictable behavior when model contracts or inference formats change. *Use Case:* A sentiment-analysis model originally designed for customer-support triage is silently reused by a marketing analytics team. When the model is updated with a new tokenization scheme, the downstream system silently begins classifying sentiment incorrectly, altering business reports. *Justification (Why):* The debt originates from undocumented and unmanaged model interfaces, making it fundamentally a model/code-level integration debt rather than a design or architectural issue.
Source(s): [26, 42, 164, 178, 213, 215].

iii. **Hidden Feedback Loops:** This occurs when model outputs unintentionally influence their future inputs, creating self-reinforcing behavior. Common examples include ranking systems, recommender pipelines, and content-moderation models, where predictions directly affect the distribution of new training data. *Impact:* These loops amplify bias, degrade robustness, and may lead to degenerative behavior (e.g., popularity bias or echo-chamber effects). Additionally, they complicate retraining, making it difficult to disentangle learned behavior from past model outputs. *Use Case:* A recommender system suggests articles to users based on previous

engagement. Over time, the system continues presenting similar content, narrowing topic diversity and reducing exploratory signals in training data. *Justification (Why):* Because feedback loops manifest through runtime coupling between model outputs and model inputs, they are inherently rooted in model-code interactions rather than algorithm design or pipeline configuration.
Source(s) [19, 106, 135, 138, 168, 178, 186, 189, 213].

iv. **Deep God File (DG):** This denotes excessively large or monolithic files—often originating from early experimentation—that contain intertwined logic for data preprocessing, model training, evaluation, and deployment [74, 164]. Such files hinder modularization, reuse, debugging, and testing. *Impact:* They reduce maintainability, complicate onboarding, hinder automated testing, and increase the likelihood of introducing defects during refactoring. *Use Case:* A Jupyter notebook used for initial research is promoted directly into production. It contains data cleaning, feature engineering, model training, and inference logic in a single script, making any modification risky. Another example is an AI application where data preprocessing, model training, and testing logic are all implemented within the same file, leading to a cluttered codebase. *Justification (Why):* DG arises from poor code organization, a foundational model-implementation issue, and is therefore grouped under Model & Code–Related Debts. [1]
Source(s): [53, 164].

v. **Prototype Debt/Dead Experimental Code Paths:** Prototype Debt accumulates when experimental prototypes or proof-of-concept scripts transition into production without refinement, optimization, or appropriate architectural restructuring. While common in AI research settings, such code is fragile and poorly suited for operational use. *Impact:* It leads to runtime inconsistencies, missing error handling, reduced scalability, and challenges in debugging or extending the model. *Use Case:* A prototype hyperparameter-tuning script is used as the production training pipeline. When data volume increases, the script fails due to hard-coded assumptions and lack of batching logic. *Justification (Why):* Its origin lies in moving research artifacts directly into production, making it fundamentally a model/code-related implementation debt.
Source(s): [26, 116–118, 138, 148, 155, 178, 204, 213, 215].

vi. **Entanglement:** It arises when multiple AI components or feature pathways become tightly interwoven, such that modifying one element requires changing others. This reduces modularity and makes isolated improvements or debugging nearly impossible. *Impact:* It creates high modification cost, restricts model evolution, and slows experimentation cycles. Errors propagate more easily, and testing becomes more complex. *Use Case:* A set of models share a common feature vector generated through a single, monolithic script. Updating any part of the vector requires revisiting all dependent models. *Justification (Why):* Entanglement reflects poor separation of concerns within model code, justifying its placement within this category.
Source(s): [26, 135, 178, 213].

vii. **Multiple Language Smells (MLS):** MLS occurs when a system incorporates multiple programming languages or framework ecosystems without clear boundaries or tooling support. Though sometimes necessary for performance (e.g., C++ extensions), unmanaged heterogeneity creates build-time and runtime fragility. *Impact:* It

[1] It is important to note that Deep God File extends the traditional God Class concept to AI contexts, where large monolithic scripts or notebooks combine data preprocessing, model design, training, and evaluation in a single file. Unlike the classic design-level anti-pattern, this debt arises from experimental ML workflows and the lack of modularization typical in AI development. Furthermore, the key difference is the motivation. A classic God Class often results from a lack of architectural discipline. In ML, the "Deep God File" often arises naturally from the exploratory nature of the work, where experimentation takes priority over software engineering best practices. The focus is on getting a working model, not on creating reusable and modular components.

increases integration complexity, complicates containerization, introduces serialization challenges, and hinders collaborative development. It also adds complexity and requires diverse expertise for maintenance. *Use Case:* A deep-learning team mixes PyTorch (Python), CUDA kernels (C++/CUDA), and custom Java-based serving infrastructure. Differences in serialization formats cause inference failures. *Justification (Why):* MLS is rooted in cross-language integration problems that directly affect model code and runtime interactions.
Source(s): [7, 138, 164, 178, 204, 213, 215].

viii. **Duplicate Model Code:** This debt occurs when identical or near-identical model functions, utilities, or architectural components are replicated across codebases or modules. It often results from parallel experimentation or copy-paste development practices. *Impact:* Leads to inconsistent behavior, duplicate bugs, and increased maintenance effort. Updating a single model component requires tracking and modifying multiple copies. *Use Case:* Two research teams independently copy the same LSTM layer implementation and introduce different small changes, resulting in inconsistent behavior across models. Redundant model code across various analytics models within an organization, requiring changes to be made in multiple locations. *Justification (Why):* The core issue—replication of model logic across code—is a model/code-level maintainability debt.
Source(s): [9, 100, 105, 116–118, 148, 155, 204, 210].

ix. **Unwanted Debugging Code (UDC):** UDC refers to temporary print statements, debug logs, or ad-hoc instrumentation left in production code. In AI systems, where inputs often include sensitive data, such artifacts may inadvertently expose information or distort performance measurements. *Impact:* It degrades performance, increases log noise, risks leaking confidential information, and complicates monitoring pipelines. *Use Case:* Residual debug logs unintentionally record user-provided medical symptoms in server logs, violating privacy obligations. Temporary code snippets used to debug a recommendation engine that, if left in production, could expose sensitive data paths. *Justification (Why):* This debt arises from failure to clean up experimental instrumentation, clearly falling within model and code practices.
Source(s): [117, 118, 148, 155, 164].

x. **Glue Code (GC):** GC refers to fragile, hand-crafted code used to connect components that were not designed to interoperate. In AI systems, this often appears in bridging incompatible frameworks, data structures, or inference formats. This debt arises when large amounts of code are written specifically to integrate libraries or components that were not initially designed to work together. Glue code often lacks proper organization, which complicates future maintenance. *Impact:* It increases brittleness, reduces portability, complicates debugging, and introduces hidden dependencies that hinder scaling and refactoring. *Use Case:* Developers write custom JSON converters to translate between two ML microservices instead of using a shared schema or API contract. *Justification (Why):* GC is inherently a code-level integration shortcut, belonging naturally to the Model & Code–Related Debts category.
Source(s): [7, 19, 25, 26, 51, 114, 117, 118, 138, 148, 155, 164, 178, 186, 204, 210, 213, 215].

All debts in this category arise from implementation-level deficiencies—including poor modularization, ad-hoc experimentation, fragile integrations, runtime coupling, and unmanaged code evolution. While these debts are not exclusive to AI-based systems and may also occur in traditional software, they are particularly prevalent and impactful in AI-enabled systems due to their tight coupling between data, models, and code. As a result, these debts weaken maintainability, increase operational risk, and hinder the scalability of AI-based systems

4.1.3 Algorithm-Related Debts. These arise from deficiencies in the conceptual design, ethical alignment, optimization objectives, or evaluation strategies of AI models. Unlike Model & Code–Related Debts, which stem from implementation and structural issues, Algorithm-Related Debts originate in the fundamental reasoning logic encoded in algorithmic choices, objective functions, and evaluation protocols. These debts influence how AI systems learn, generalize, and behave under real-world conditions, often with long-term consequences for fairness, explainability, robustness, and trustworthiness.

Because algorithmic decisions shape the system's predictive behavior and risk profile, debts in this category propagate into downstream models, user interactions, and sociotechnical ecosystems. The following debts fall under this category.

i. **Algorithm Inclination Debt (Human Bias Debt):** Occurs when bias is inadvertently embedded within the model design, objective functions, or feature selection strategies. Unlike Data Debt, which concerns bias in the training corpus, this debt reflects bias introduced by model developers—through choices such as skewed class weights, inappropriate regularization terms, or the omission of fairness-aware modeling techniques. These decisions influence who benefits or is disadvantaged by the model's outputs. It also refers to a bias introduced by over-reliance on familiar algorithms, even when they are suboptimal for the problem. *Impact:* It leads to discriminatory outcomes, lower fairness, undermined user trust, and legal or ethical liabilities. Such bias often snowballs in production systems, particularly in high-stakes applications such as credit scoring, hiring, policing, and healthcare. *Use Case:* A loan-approval classifier is optimized solely for overall accuracy on an imbalanced dataset. Because the cost of false negatives is not penalized for minority applicants, the model disproportionately rejects their applications—a consequence stemming from the unfair optimization objective, not the raw data itself. A recommendation system using outdated algorithms that fail to adapt to user behavior, causing dissatisfaction. *Justification (Why):* This debt is categorized under Algorithm-Related Debts because the root cause is the algorithmic formulation—not code quality or data defects. It emerges from modeling choices, loss-function configuration, class balancing strategies, and inductive biases directly encoded into the learning process.
Source(s): [19, 42, 44, 60, 116, 120, 121, 146, 190, 213].
ii. **Ethical Debt:** Ethical Debt accumulates when the development process neglects moral, societal, or human-centric considerations—such as user consent, privacy preservation, transparency, or fairness. Ethical Debt differs from Algorithm Inclination Debt in that it encompasses broader governance-level harms beyond algorithmic optimization, including neglect of ethical guidelines, insufficient documentation of value trade-offs, and absence of bias-impact assessments. *Impact:* It increases the risk of harmful or discriminatory system behavior, erodes public trust, exposes organizations to regulatory penalties, and can lead to reputational damage. Ethical Debt often becomes visible only after deployment when real-world impacts emerge. *Use Case:* A facial-recognition model is deployed in a public environment without transparency documentation or ethical risk analysis. Later, it is found to perform poorly on underrepresented demographic groups, leading to public backlash and regulatory scrutiny. A healthcare AI system that exhibits racial bias due to lack of diverse data, resulting in unequal treatment recommendations. *Justification (Why):* Ethical Debt belongs to this category because its origin is algorithmic governance and moral oversight. It reflects decisions made at the conceptual design and alignment stage, rather than issues of code structure or operational infrastructure.
Source(s): [40, 135, 158, 168].
iii. **Overly Simplified Metrics:** refers to the use of oversimplified, misaligned, or insufficient evaluation metrics that fail to capture the real-world performance, safety, or societal impacts of AI models. Focusing solely on metrics such

as accuracy, precision, or F1-score may obscure system vulnerabilities—such as sensitivity to out-of-distribution data, robustness under adversarial conditions, or fairness disparities. It can also refer to the use of metrics that are generic or misaligned with specific problem requirements. *Impact:* This debt leads to misleading validation results, a false sense of model reliability, and unanticipated failures when deployed in dynamic environments. Models optimized for incomplete or Overly Simplified Metrics may perform well in controlled experiments but fail dramatically in real-world contexts. *Use Case:* A medical-diagnosis model is evaluated exclusively using overall accuracy. Rare but critical conditions (minority classes) are under-detected because the metric does not capture the cost of false negatives. Deployment then results in missed diagnoses and potential patient harm. *Justification (Why):* The root cause lies in evaluation-design inadequacy, an algorithmic-level failure to define meaningful and context-aware success metrics. Therefore, Overly Simplified Metrics Debt is rightfully placed under Algorithm-Related Debts rather than testing or data categories.
Source(s): [42, 105, 220].

Algorithm-Related Debts arise from conceptual design choices that govern how AI systems learn, reason, and behave. These debts are epistemic rather than structural—stemming from flawed optimization objectives, inadequate evaluation strategies, or ethical misalignment rather than software engineering deficiencies. If left unaddressed, they compromise fairness, safety, trustworthiness, and long-term societal impact, making them critical to responsible AI development and governance.

4.1.4 Design/Architecture Debts. These debts arise from deficiencies in the structural organization, interface boundaries, and configuration management of AI-based systems. These debts reflect weaknesses in how system components are designed, how architectural layers interact, and how configurations are specified, tracked, and maintained. Unlike Model & Code–Related Debts, which emerge from implementation artifacts, these debts originate earlier in the system's conceptual and structural design. They compromise long-term extensibility, maintainability, and reproducibility—attributes especially critical in AI systems, where models depend on complex pipelines, configurable hyperparameters, and evolving architectural patterns. The debts in this category collectively undermine internal cohesion, architectural integrity, and configuration reliability.

i. **Jumbled Model Architecture (JMA):** JMA arises when AI models—particularly deep learning architectures—lack coherent structural organization. Examples include inconsistent layering schemes, irregular activation patterns, or ad-hoc combinations of architectural modules. Such models often originate from rapid experimentation or incremental patching during prototyping. *Impact:* It reduces interpretability, increases training instability, complicates debugging, and diminishes the model's ability to generalize. Since architecture heavily influences representational learning, fragmented structures can introduce vanishing gradients, bottlenecks, or redundant paths. *Use Case:* A computer vision team alternates between different convolutional block types across layers due to experimentation. The resulting architecture exhibits sporadic gradient explosions during training, making optimization highly unstable; An AI system developed with a mix of different neural network architectures without clear separation between components, leading to difficulties in maintenance. *Justification (Why):* JMA belongs to this category because it reflects a design-phase structural deficiency rather than implementation or code-level issues. The root cause is architectural inconsistency originating from poor design discipline.
Source(s): [9, 53, 69, 116, 117, 155, 156, 164, 175].

ii. **Boundary Erosion:** This occurs when architectural boundaries—such as module interfaces, layer abstractions, or service contracts—are violated. In AI systems, this may manifest when data preprocessing modules directly

interact with model internals, or when models bypass APIs to access raw system components. Over time, the boundaries between components in an AI system may erode due to complex interdependencies, weakening the modularity of the system and increasing maintenance difficulty. *Impact:* It increases coupling, reduces modularity, and exposes internal components to misuse. Eroded boundaries introduce security risks, complicate dependency management, and restrict the ability to update or replace components independently. *Use Case:* In a multi-model AI platform, as models become interdependent, changes in one affect others, eroding the clear separation between components; A feature-engineering script directly injects intermediate tensors into a model's internal layers for debugging. Later changes to the model break the pipeline, as external components were never meant to interact with these layers; *Justification (Why):* Boundary Erosion's root cause is architectural-layer violation, making it conceptually distinct from code smells or operational issues and thus appropriately classified here.
Source(s): [42, 178, 204].

iii. **AI Architectural Debt** Refers to architectural or structural decisions that trade long-term quality for short-term speed, simplicity, or expedience. In AI systems, this often appears as simplistic model wrappers, hard-coded hyperparameters, or missing abstraction layers that prevent future scaling or generalization. *Impact:* It limits reusability, slows feature development, and introduces fragility into evolving codebases. Poor design also impedes systematic monitoring, experimentation, and integration—key aspects of continuous AI deployment. *Use Case:* A research team wraps a model inside a single script without separating preprocessing, inference, and monitoring logic. As production requirements evolve, each modification forces large-scale refactoring. *Justification (Why):* Design Debt is inherently a structural design failure, justifying its assignment to this category rather than implementation or operational domains.
Source(s): [9, 25, 60, 100, 116–118, 120, 121, 146, 155, 203, 222].

iv. **Configuration Debt:** This debt arises when hyperparameters, environment variables, or system settings are poorly documented, inconsistently defined, or not version-controlled. In AI systems—where configuration governs behavior as much as code—such inconsistencies cause unpredictability and hinder reproducibility. Over time, configuration settings in an AI system become overly complex or difficult to manage, increasing the risk of errors and misconfiguration. *Impact:* It produces experimental inconsistencies, failed model replications, training divergence, or mismatched behavior between training and inference. Configuration Debt frequently leads to irreproducible results, a central challenge in machine learning research. *Use Case:* A deep learning model fails to reproduce validation accuracy because random seeds, batch sizes, and learning schedules were not logged during earlier experiments; An AI system with numerous hyperparameter settings, making it difficult for operators to manage configurations accurately across deployment environments. *Justification (Why):* This debt reflects configurational mismanagement, a structural rather than implementation issue, and is therefore appropriately grouped under this category.
Source(s): [7, 26, 32, 44, 69, 100, 105, 146, 178, 204, 220, 224].

v. **Compatibility Debt:** This emerges when AI components rely on outdated, incompatible, or conflicting frameworks, model formats, or serialization schemes. This often occurs when transitioning between versions of TensorFlow, PyTorch, ONNX, or other tooling ecosystems. *Impact:* It hinders system evolution, complicates deployment across environments, increases integration overhead, and may prevent models from being exported or reused across platforms; Incompatibilities among system components lead to inefficient workarounds, raising system complexity and maintenance challenges. *Use Case:* A model trained in an older TensorFlow version cannot be exported to ONNX without extensive patching, delaying migration to a faster inference engine. *Justification*

(Why): Its origin lies in architectural and design decisions regarding toolchain selection, framework versions, and dependency constraints—not in code-level implementations.
Source(s): [44, 114, 120, 121].

vi. **Abstraction Debt:** Occurs when interfaces are overly generic, insufficiently defined, or inconsistently structured. Poor abstraction design leads to convoluted inheritance hierarchies, unclear responsibilities, and tightly coupled components. *Impact:* It makes model components harder to extend, test, and reuse. Abstraction Debt often introduces hidden coupling and forces developers to modify low-level details even for high-level changes. *Use Case:* A base "Model" class defines ambiguous abstract methods, causing inconsistent implementations across subclasses. Adding a new training mode requires modifying nearly all subclassed models. *Justification (Why):* This debt's root cause is inadequate abstraction in design, making it a prototypical structural debt falling under Design, Architecture & Configuration.
Source(s): [51, 178, 204, 215].

Design, Architecture & Configuration Debts reflect structural and conceptual weaknesses introduced during the design and configuration stages of AI system development. These debts degrade modularity, hinder reproducibility, and amplify technical friction across the lifecycle. They differ from implementation-level debts because their origins lie in poor architectural foresight, inconsistent configuration practices, and inadequate abstraction mechanisms rather than the code itself.

4.1.5 Operational & Lifecycle Debts. These emerge from weaknesses in the processes, infrastructure, and versioning practices that support the continuous development, deployment, and maintenance of AI systems. Unlike debts rooted in design or implementation, these reflect failures in operational discipline and lifecycle governance, including inadequate CI/CD automation, untracked evolution of model artifacts, and ambiguous requirements that lead to rework. Because AI systems require tight synchronization across model, data, and environment versions, even routine operational lapses can manifest as long-term technical liabilities, particularly in production environments.

i. **Requirement Debt:** This accumulates when functional or quality attribute specifications are ambiguous, incomplete, or inconsistently communicated [62]. AI systems are particularly vulnerable because their performance depends on well-defined metrics, target thresholds, and operational constraints. Poor requirement alignment often leads to misinterpretation of expected outcomes or inappropriate model objectives. *Impact:* This debt results in rework, misaligned performance goals, wasted experimentation cycles, and delayed delivery. It also increases the risk of deploying models that fail to meet stakeholder expectations or regulatory requirements. *Use Case:* A healthcare model is trained to maximize accuracy, but stakeholders later clarify that minimizing false negatives is critical for patient safety. The model must be retrained, causing significant delays; An AI-based e-commerce system where user experience requirements were insufficiently defined, leading to an incomplete recommendation feature. *Justification (Why):* Because the root cause is inadequate requirements engineering within the lifecycle, Requirements Debt is classified as an Operational & Lifecycle Debt rather than an algorithmic or design-level issue.
Source(s): [9, 25, 26, 60, 116–118, 120, 121, 138, 142, 148, 155, 203].

ii. **Defect Debt:** This arises when known bugs or system flaws are deferred for future resolution, often due to time pressure or resource limitations. These defects may include incorrect preprocessing logic, mislabeled data, unstable model behaviors, or unhandled exceptions. *Impact:* It reduces system reliability, increases the frequency of runtime failures, and demands greater developer time to diagnose and correct issues later in the lifecycle. *Use*

Case: A feature extractor intermittently outputs NaN values during training. The issue is noted but deprioritized. Over time, the intermittency increases, leading to complete training failure during a critical deployment phase. *Justification (Why):* The root cause is post-development defect tolerance, an operational choice that directly affects lifecycle maintenance.
Source(s): [9, 25, 116, 117, 120, 121, 148, 155, 222].

iii. **Versioning Debt:** This occurs when model artifacts, datasets, code modules, or environment configurations are not appropriately version-controlled. AI systems heavily depend on deterministic versioning to ensure reproducibility, auditability, and rollback capability. Poor version control practices lead to challenges in tracking changes, maintaining compatibility, and reproducing past results. *Impact:* It obstructs experiment tracking, prevents reproducibility of results, hinders debugging, and complicates compliance audits. Models may behave unpredictably because different environment or dataset versions inadvertently enter the pipeline. *Use Case:* A new model underperforms during A/B testing. Investigation reveals that the training dataset version differs from the dataset used in validation, but no version metadata was tracked. *Justification (Why):* Because the root cause lies in lifecycle artifact management, this debt belongs to the Operational & Lifecycle category.
Source(s): [9, 52, 155, 189, 215].

iv. **Process/Infrastructure Debt:** This debt arises from outdated, inconsistent, or manual operational workflows, including CI/CD pipelines, orchestration processes, hardware provisioning, or monitoring infrastructure. AI lifecycles often rely on complex tooling (e.g., model registries, deployment orchestrators, monitoring dashboards), making process deficiencies especially harmful. *Impact:* It slows deployment cycles, increases operational risk, reduces system scalability, and introduces failure points due to manual interventions. *Use Case:* A model deployment pipeline requires manual execution of several scripts. When an urgent patch is needed, a step is executed incorrectly, causing service downtime; An AI-based recommendation system with an outdated build pipeline lacking automation. *Justification (Why):* Since the underlying issue is lack of automation and process maturity, it is appropriately placed under Operational & Lifecycle Debts.
Source(s): [114, 142, 155].

v. **Build Debt:** This accumulates when build scripts, Dockerfiles, environment specifications, or compilation pipelines become fragmented, non-standardized, or poorly maintained. This often results from ad-hoc experimentation during early development stages. *Impact:* It causes inconsistent builds, extends debugging time, creates deployment fragility, and increases onboarding complexity. Build Debt is a common source of environment drift between development, testing, and production; Results in unreliable deployment processes and system instability. *Use Case:* A production inference container fails because a dependency version differs from development. The mismatch originates from an outdated Dockerfile with ambiguous version specifications. *Justification (Why):* The root cause is build environment instability, making this a lifecycle and operational concern rather than a code-level one.
Source(s): [116, 117, 155].

These debts all stem from operational misalignment, inadequate automation, and poor lifecycle management practices. If unaddressed, they significantly undermine the reliability, reproducibility, and agility of AI development and deployment.

4.1.6 Documentation and Communication Debt. These debts originate from insufficient documentation, weak communication channels, and inadequate knowledge transfer practices across teams. AI systems rely on complex interactions between data engineers, ML researchers, software developers, and domain experts. When documentation is incomplete

or communication is fragmented, system behavior becomes opaque, maintenance becomes slower, and organizational risk increases. These debts reflect socio-technical deficiencies rather than purely technical ones.

i. **Documentation Debt:** This arises when key artifacts—such as model behavior, preprocessing steps, data assumptions, evaluation methodologies, or architectural decisions—are insufficiently documented, outdated, or missing entirely [62]. *Impact:* It slows onboarding, increases maintenance effort, complicates audits, and leads to miscommunication during handovers. For AI systems, lack of documentation also undermines explainability, transparency, and regulatory compliance; Poor documentation creates obstacles for developers to maintain and extend AI systems effectively, resulting in reduced system reliability. *Use Case:* A deployment engineer attempts to reproduce a training run but cannot locate information on normalization strategies used in the original experiment, resulting in inconsistent model behavior; Sparse documentation in a complex AI model for disease diagnosis, which complicates onboarding new developers and troubleshooting. *Justification (Why):* This debt's root cause is deficiency in documentation and knowledge management, making it a socio-technical rather than algorithmic or architectural debt.
Source(s): [9, 25, 40, 52, 94, 116–118, 120, 121, 142, 148, 155, 203, 213].

ii. **Cultural/People/Social Debt:** This debt arises when teams have misaligned communication practices, insufficient collaboration, or conflicting expectations. In AI settings, teams often operate in silos—data engineers manage schemas, ML researchers iterate on models, and software engineers maintain deployments—causing integration friction. It also refers to the differences in culture, values, or priorities between teams, such as research and engineering, leading to misalignments [62]. *Impact:* It leads to coordination failures, inconsistent workflows, misconfigured models, duplicated effort, and erroneous assumptions about system behavior. It also increases risk in safety-critical or regulated contexts. *Use Case:* An ML team updates a model's input schema but does not communicate the change to the data engineering team. As a result, the data pipeline continues to produce the old schema, causing runtime failures and misaligned features. *Justification (Why):* The root cause is organizational misalignment, not technical implementation defects. Thus, it aligns naturally with Documentation & Communication Debts.
Source(s): [4, 16, 19, 60, 114, 129, 135, 138, 142, 155].

Documentation & Communication Debts highlight the human and organizational dimension of technical debt. These debts impair transparency, collaboration, and maintainability, especially in interdisciplinary AI teams where knowledge silos are common.

4.1.7 Testing and Quality Assurance Debt. These debts emerge from insufficient verification, limited test coverage, and overlooked quality-assurance practices. AI systems require testing not only of code correctness but also of data pipelines, model behavior under distribution shifts, fairness metrics, and robustness to adversarial conditions. Inadequate testing accelerates technical debt accumulation, reduces trustworthiness, and increases the likelihood of harmful failures in production.

i. **Test Debt:** Test Debt accumulates when testing suites are incomplete, outdated, or insufficient to assess AI system behavior. This includes lack of unit tests for preprocessing logic, missing integration tests, absence of fairness or robustness evaluations, and insufficient testing of edge-case inputs [62]. *Impact:* It leads to undetected regressions, silent model degradation, vulnerability to distribution shifts, and reduced confidence in deployments. Test Debt often manifests only when unexpected failures occur in real-world usage. *Use Case:* A sentiment-analysis model

fails to handle emerging slang and social-media phrases because its test suite only covers standard dictionary terms. A predictive analytics model deployed without comprehensive testing, resulting in undetected errors in real-time use. *Justification (Why):* The debt originates from insufficient QA practices, making it distinct from data, architecture, and operational deficiencies.
Source(s): [9, 19, 25, 32, 60, 114, 116–118, 120, 121, 148, 155, 186, 203, 204, 213].

RQ1.1: The analysis revealed that AI Technical Debts (AITDs) are best explained through a root-cause–oriented perspective comprising seven overarching categories: (1) Data & Library-Related Debts, (2) Model & Code-Related Debts, (3) Algorithm-Related Debts, (4) Design, Architecture & Configuration Debts, (5) Operational & Lifecycle Debts, (6) Documentation & Communication Debts, and (7) Testing & Quality Assuarance Debts. By grounding the taxonomy in the underlying engineering sources—such as data instability, structural design flaws, algorithmic misalignment, inadequate lifecycle processes, and socio-technical communication gaps—the classification provides a holistic and integrative view of the diverse mechanisms through which technical debt accumulates in AI-based systems.

RQ1.2: Through grounded theory analysis, we identified and refined a total of 31 distinct AITDs reported across the 60 primary studies. These debts span all seven root-cause categories and collectively reflect the multifaceted nature of technical debt in AI-enabled systems. Their prevalence varies considerably across the literature, with *Data Debt (45.00%)*, *Glue Code (30.00%)*, and *Test Debt (28.33%)* emerging as the most frequently reported issues. Table 4 summarizes the occurrence frequencies and provides a detailed overview of how often each debt type appears across empirical studies.

4.2 Analysis of the Identified AITDs and Their Implications Through the Lens of AI TRiSM

Table 4 summarizes the 31 AI Technical Debts (AITDs) identified across the 60 primary studies and ranks them by frequency and percentage occurrence. The distribution reveals several dominant forms of debt that consistently appear across diverse AI system development contexts.

Data Debt is the most frequently reported AITD, appearing in **45.00%** of the studies. This debt category reflects persistent data governance and data-quality challenges, including unstable data dependencies, schema inconsistencies, missing metadata, and untracked transformations. Given that AI systems rely on continuously evolving and heterogeneous data sources, these deficiencies lead to model drift, silent performance degradation, and reproducibility failures. Data Debt remains the most systemic and foundational problem recorded in the literature.

Glue Code is the second-most prevalent, reported in **30.00%** of the studies. It captures the ad-hoc integration logic used to connect disparate data pipelines, libraries, model components, and deployment tools. Although Glue Code enables rapid experimentation—particularly in early development phases—it produces fragile, highly coupled systems that are difficult to extend or debug. This debt reflects the rapid, tool-centric, and exploratory nature of typical AI workflows.

Test Debt ranks third, appearing in **28.33%** of the included studies. AI-based systems require extensive testing not only at the code level but also across data transformations, model behavior, fairness, robustness, and performance under distribution shift. The literature indicates that teams frequently deprioritize testing due to the difficulty of specifying

Table 4. Overview of AI Technical Debts (AITDs) as part of the systematic review

#	AITD	Freq(n=60)	%	Category
1	Data Debt/Unstable Data Dependencies	27	45.00	Data/Library Related Debts
2	Glue Code (GC)	18	30.00	Model/Code Related Debts
3	Test Debt	17	28.33	Testing & Quality Assurance Debts
4	Documentation Debt	15	25.33	Documentation & Communication Debts
5	Requirement Debt	14	23.33	Operational & Lifecycle Debts
6	AI Architectural Debt	13	21.67	Design & Architecture Debts
7	Configuration Debt	12	20.00	Design & Architecture Debts
8	Dead Experimental Code Paths/Prototype Debt	11	18.33	Model/Code Related Debts
9	Algorithm Debt/Inclination/Human Bias	10	16.67	Algorithm Related Debts
10	Duplicate Model Code	10	16.67	Model/Code Related Debts
11	Cultural/People/Social Debt	10	16.67	Documentation & Communication Debts
12	Pipeline Jungle	9	15.00	Data/Library Related Debts
13	Jumbled Model Architecture (JMA)	9	15.00	Design & Architecture Debts
14	Hidden Feedback Loops	9	15.00	Model/Code Related Debts
15	Defect Debt	9	15.00	Operational & Lifecycle Debts
16	Multiple Language Smells (MLS)	7	11.67	Model/Code Related Debts
17	Undeclared consumers	6	10.00	Model/Code Related Debts
18	Ethical Debt	6	10.00	Algorithm Related Debts
19	Unwanted Debugging Code (UDC)	5	8.33	Model/Code Related Debts
20	Versioning Debt	5	8.33	Operational & Lifecycle Debts
21	Correction Cascades (CC)	4	6.67	Model/Code Related Debts
22	Entanglement	4	6.67	Model/Code Related Debts
23	Compatibility Debt	4	6.67	Design & Architecture Debts
24	Abstraction Debt	4	6.67	Design & Architecture Debts
25	Dispensable Dependency	3	5.00	Data/Library Related Debts
26	Boundary Erosion	3	5.00	Design & Architecture Debts
27	Overly Simplified Metrics	3	5.00	Algorithm Related Debts
28	Process/Infrastructure Debt	3	5.00	Operational & Lifecycle Debts
29	Build Debt	3	5.00	Operational & Lifecycle Debts
30	Scattered Use of ML Libraries (SML)	2	3.33	Data/Library Related Debts
31	Deep God File (DG)	2	3.33	Model/Code Related Debts

expected ML behavior or the iterative nature of model development. This leads to insufficient test coverage, poor monitoring, and higher risk of failure in deployment settings.

Documentation Debt follows closely at **25.33%**. As AI projects involve interdisciplinary teams (AI researchers, ML engineers, domain experts), inadequate documentation compromises maintainability, reproducibility, and onboarding. Missing or outdated dataset descriptions, model cards, configuration files, feature-engineering rationales, and pipeline diagrams hinder long-term system understanding and evolution.

Requirement Debt appears in **23.33%** of the studies, illustrating how evolving stakeholder needs and ambiguous system goals affect AI development. Requirement Debt often arises when model behavior is not fully specified or when performance and fairness requirements evolve during experimentation. This leads to incomplete implementations and rework.

AI Architectural Debt and **Configuration Debt** were reported in **21.67%** and **20.00%** of the papers respectively. Design Debt arises from rushed architectural decisions and insufficient modularity—often exacerbated by early experimentation culture. Configuration Debt reflects the complexity of hyperparameters, environment variables, orchestration specifications, and deployment parameters, which, when poorly managed, cause reproducibility gaps and environment-dependent failures.

Prototype Debtappear in **18.33%** of the studies. Prototype Debt stems from experimental code that is prematurely reused in production. It illustrate the tension between research-driven iteration and production engineering.

Following these, a second tier of moderately frequent debts (reported in 15–17% of studies) includes: *Algorithm Inclination/Human Bias (16.67%)*, *Duplicate Model Code (16.67%)*, *Cultural/People/Social Debt (16.67%)*, *Pipeline Jungle (15.00%)*, *Jumbled Model Architecture (JMA) (15.00%)*, *Hidden Feedback Loops (15.00%)*, *Defect Debt (15.00%)*, *Multiple Language Smells (MLS) (11.67%)*, *Undeclared Consumers (10.00%)*, *Ethical Debt (10.00%)*. These debts highlight structural, behavioral, and socio-technical issues such as unplanned architecture evolution (JMA), unintended system dynamics (Hidden Feedback Loops), implementation redundancy (Duplicate Model Code), ethically problematic decision pathways (Ethical Debt), and misalignment across teams (Cultural/People/Social Debt).

The remaining AITDs, each appearing in fewer than *10%* of the studies, include Unwanted Debugging Code (8.33%), Versioning Debt (8.33%), Correction Cascades (6.67%), Entanglement (6.67%), Compatibility Debt (6.67%), Abstraction Debt (6.67%), Dispensable Dependency (5.00%), Boundary Erosion (5.00%), Overly Simplified Metrics (5.00%), Process/Infrastructure Debt (5.00%), Build Debt (5.00%), Scattered Use of ML Libraries (3.33%), and Deep God File (3.33%).

Although individually less frequent, these debts are still significant. Many reflect emerging or specialized risks unique to AI complexity, such as: unstable component borders (Boundary Erosion), cascading dependency breakages (Correction Cascades), flawed or manipulated evaluation criteria (Overly Simplified Metrics), architectural monoliths (Deep God File), tool fragmentation (Scattered ML Libraries). Their lower frequency does not imply low severity; several represent high-impact failure modes that are under-investigated in current research.

The prevalence distribution reinforces the inherently multifaceted nature of AITDs. High-impact debts (Data Debt, Glue Code, Test Debt) correlate strongly with foundational AI engineering workflows—data management, model integration, and verification—while medium-frequency and low-frequency debts highlight structural, algorithmic, operational, and socio-technical weaknesses. In summary, the landscape of AITDs reveals a complex interplay between AI-specific engineering challenges and long-standing software quality concerns. The prevalence analysis indicates that AI-intensive systems demand specialized management strategies addressing data governance, integration practices, rigorous testing, reproducibility, and responsible algorithmic design to ensure sustainable and trustworthy AI development.

Implications of Prevalent AITDs Through the Lens of AI TRiSM. Taken together, the ten most prevalent AITDs identified in this review—Data Debt/Unstable Data Dependencies (Data/Library-Related Debts); Glue Code, Prototype Debt, and Duplicate Model Code (Model/Code-Related Debts); Test Debt (Testing & Quality Assurance Debts); Documentation Debt (Documentation & Communication Debts); Requirement Debt (Operational & Lifecycle Debts); AI Architectural Debt and Configuration Debt (Design & Architecture Debts); and Algorithmic Inclination/Human Bias (Algorithm-Related Debts)—exhibit clear, mechanism-level intersections with AI safety and AI security concerns.

Data and library–related weaknesses (e.g., unstable data dependencies) increase the likelihood of unsafe behavior under data drift and reduce confidence in input validity. Model and code–related debts (e.g., glue code, duplicate model code, and prototype paths) introduce fragile integration layers and hidden dependencies that broaden attack surfaces and make failures harder to isolate. Testing and documentation deficits reduce fault and vulnerability discoverability and

weaken auditability and incident response capabilities, while operational and architectural debts—such as requirements ambiguity, configuration drift, and architectural shortcuts—can embed unsafe assumptions, misconfigurations, and weak boundary controls into production systems. Finally, algorithm-related bias debt directly contributes to harmful outcomes and safety violations through systematic misbehavior across specific populations or operational contexts.

This analysis demonstrates how the most frequently reported AITDs consistently map to concrete safety- and security-relevant risk pathways, thereby supporting an AI-TRiSM–aligned perspective on AITD management. This mapping, however, should be regarded as an initial analytical step; Sections 5 and 6 provide a deeper examination of how these AITDs propagate and compound risks related to security and safety, as well as their management -the foundational pillars of AI TRiSM—thereby directly addressing **RQ2–RQ3**.

5 AITDs - Safety and Security Concerns (RQ2)

5.1 Safety Concerns

The increasing integration of AI in high-stakes domains has amplified the need to address critical safety concerns that extend beyond technical debt and performance limitations. As detailed in prior AI safety literature, particularly in [33, 84, 162, 171], key safety risks emerge due to the opaque, non-deterministic, and dynamic nature of AI systems. These concerns are especially pronounced in systems leveraging deep learning and reinforcement learning models, which are often sensitive to environmental noise, distributional shifts, and adversarial manipulation. Table 5 presents a comprehensive mapping between these safety concerns and their related AITDs, providing an explanatory context for each relationship.

5.1.1 Explainability and Interpretability. AI models - particularly deep learning architectures - often operate as black boxes, making it difficult for developers and stakeholders to understand, validate, or justify system decisions. In addition to this opacity, many AI models exhibit inherently stochastic behavior due to probabilistic inference, non-deterministic training processes, and adaptive updates. Such stochasticity reduces predictability, which has traditionally been a cornerstone of safety and security engineering. Together, limited interpretability and non-deterministic behavior pose significant safety risks in domains such as healthcare, finance, and autonomous systems, where accountability, traceability, and reliable error analysis are essential [2, 8, 39, 84, 111, 183, 221].

Related AITDs:

- *Jumbled Model Architecture*: Poorly structured and undocumented model architectures obscure the internal logic of AI systems, making it difficult to interpret model behavior or trace decision flows.
- *Overly Simplified Metrics*: When models are evaluated using high-level, generic metrics (e.g., accuracy), without alignment to context-specific safety goals (e.g., false negatives in medical diagnosis), the model's performance becomes misleading and less interpretable.
- *Documentation Debt*: Lack of sufficient documentation about models, features, and training processes severely impairs the ability of users and auditors to interpret system logic, creating blind spots in safety validation efforts.

5.1.2 Robustness and Reliability. Robustness is the ability of AI systems to remain stable and accurate under perturbations, noisy data, or adversarial conditions [5, 18, 37, 47, 75, 84, 140, 205, 212, 228]. Reliability ensures consistent performance over time and across various deployment environments. A lack of robustness and reliability can cause erratic behavior and system failures, undermining trust and safety in AI operations [67, 112, 128, 177, 216, 227].

Related AITDs:

- *Data Debt*: Poor quality, inconsistent, or evolving training data leads to brittle models that fail when exposed to real-world or shifted inputs.
- *Defect Debt*: Known but unresolved bugs in the system compromise reliability, especially as they accumulate and interact unpredictably with evolving components.
- *Test Debt*: Inadequate testing, particularly in edge-case scenarios, increases the likelihood of system crashes or incorrect outputs under stress conditions.

5.1.3 Transparency and Trustworthiness. Transparency refers to the ability to inspect, understand, and verify AI processes [34, 39, 48, 111, 172, 183, 194, 227]. Trustworthiness builds upon transparency by assuring users that the system behaves ethically, consistently, and as intended [31, 39, 86, 90, 199, 200]. In safety-critical settings, lack of transparency can result in hidden vulnerabilities, while eroded trust may prevent responsible AI adoption.

Related AITDs:

- *Documentation Debt*: Insufficient documentation prevents clear understanding of system behavior, hindering transparency and the ability to audit the model pipeline.
- *Versioning Debt*: Poor version tracking of datasets, models, or parameters makes it difficult to reproduce results, undermining system transparency and regulatory traceability.
- *Ethical Debt*: Failing to explicitly consider ethics or fairness in system design leads to opacity in value alignment, undermining user trust.

5.1.4 Bias and Fairness. Bias in AI systems arises from imbalanced datasets, unrepresentative sampling, or discriminatory design choices. These biases compromise fairness and can result in systemic harm to individuals or groups, particularly in sensitive domains such as hiring, lending, and criminal justice [98, 107, 107, 199].

Related AITDs:

- *Algorithmic Inclination / Human Bias*: Over-reliance on familiar or convenient algorithms, rather than evaluating fairness implications, can encode human or institutional bias into the system.
- *Ethical Debt*: Failure to integrate ethical design principles allows biased data or assumptions to propagate unchecked, resulting in unfair or unsafe model decisions.
- *Overly Simplified Metrics*: Metrics that fail to account for fairness dimensions (e.g., equal opportunity, demographic parity) obscure potential bias and mask harmful outcomes.

5.1.5 Adversarial and Poisoning Attacks. Adversarial attacks involve input manipulations crafted to mislead AI systems, while data poisoning injects malicious data into training pipelines to degrade model integrity. These threats compromise system safety by causing incorrect outputs that may go undetected in critical environments [12, 37, 90, 127, 128].

Related AITDs:

- *Algorithmic Inclination / Human Bias*: Use of predictable or widely known algorithms increases susceptibility to adversarial manipulation, especially if security hardening is not prioritized.
- *Correction Cascades*: Inter-model dependencies make it possible for adversarial changes in one component to cascade through the system, amplifying unsafe behaviors.
- *Entanglement*: Tight coupling between components means that adversarially targeted faults in one model can trigger ripple effects, leading to systemic vulnerabilities.

5.1.6 Out-of-Distribution (OOD) Generalization. AI systems are often trained in controlled settings and may fail when deployed in real-world environments that differ from their training data distributions. OOD generalization failures can result in unsafe decisions, especially when the system is unaware of its own limitations [17, 31, 37, 75, 87, 103, 162, 169].

Related AITDs:

- *Pipeline Jungle*: Overly complex and interdependent data pipelines hinder the ability to adapt to new data distributions, leading to failures in OOD settings.
- *Dead Experimental Code Paths / Prototype Debt*: Experimental or prototype components may lack robustness and are often not validated against diverse real-world data, making them vulnerable to distribution shifts.
- *Overly Simplified Metrics*: Over-reliance on aggregate metrics during training can hide poor generalization performance on unseen inputs, leading to unsafe or unexpected system behavior post-deployment.

Table 5. Mapping Safety Concerns to Related AI Technical Debts (AITDs)

S/N	Safety Concern	Related AITDs	Explanation
1	Explainability and Interpretability	Jumbled Model Architecture, Overly Simplified Metrics, Documentation Debt	Opaque model structures, vague or misaligned evaluation metrics, and poor documentation reduce the ability to understand or justify AI decisions—critical in high-stakes scenarios like healthcare and law.
2	Robustness and Reliability	Data Debt, Defect Debt, Test Debt	Poor data quality, unresolved bugs, and insufficient testing undermine model stability and cause erratic or unreliable performance in unpredictable real-world environments.
3	Transparency and Trustworthiness	Documentation Debt, Versioning Debt, Ethical Debt	Incomplete documentation, weak version tracking, and disregard for ethical considerations hinder traceability and reduce stakeholder confidence in the system's outputs.
4	Bias and Fairness	Algorithmic Inclination / Human Bias, Ethical Debt, Overly Simplified Metrics	Systems trained on biased data or developed without ethical oversight can produce unfair or discriminatory results, particularly when using simplistic or poorly aligned performance metrics.
5	Adversarial and Poisoning Attacks	Algorithmic Inclination / Human Bias, Correction Cascades, Entanglement	Overused algorithms, coupled components, and cascading model dependencies increase the attack surface and enable faults or adversarial perturbations to propagate across the system.
6	Out-of-Distribution (OOD) Generalization	Pipeline Jungle, Dead Experimental Code Paths, Overly Simplified Metrics	Complex or disorganized pipelines, unvetted prototype code, and overly generic evaluation metrics result in models that perform poorly or dangerously when encountering unfamiliar input distributions.

To provide a visual representation of the relationships between identified safety concerns and the corresponding AI technical debts, we present Figure 4. This mapping helps to clarify how various safety attributes are undermined by specific AITDs and reinforces the interconnectedness between safety assurance and technical debt management in AI systems.

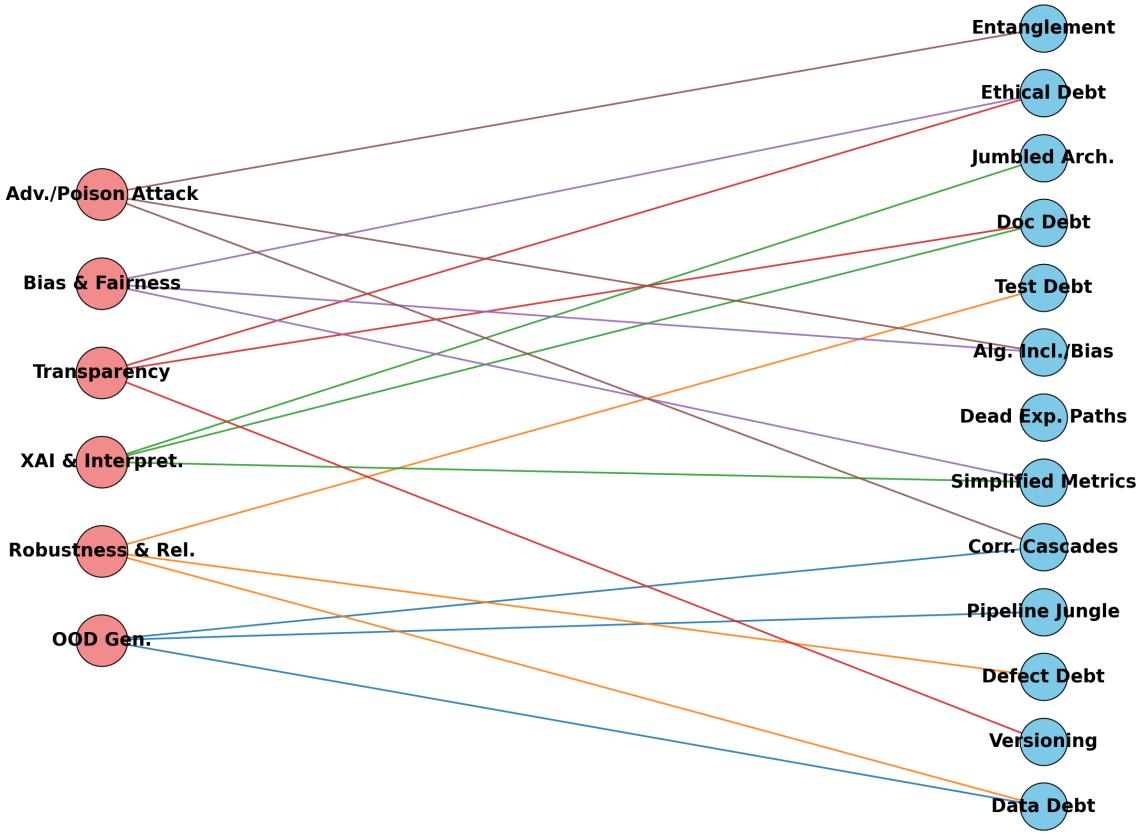

Fig. 4. **Visual Mapping of Safety Concerns to AI Technical Debts:** Red circles represent safety concerns, while blue circles represent AITDs. Each safety concern is linked to its associated technical debts using color-coded edges, illustrating the multidimensional impact of technical debt on AI system safety.

RQ2.1 – Safety Concerns in AI Systems

AI Technical Debts (AITDs) undermine the safety of AI systems in high-stakes environments. Key safety risks include:

- **Low Explainability:** Complex and poorly structured models reduce interpretability.
- **Poor Robustness:** Fragile models fail under noisy or adversarial inputs.
- **Opaque Behavior:** Documentation and versioning issues erode transparency and trust.
- **Bias Risks:** Algorithmic shortcuts and Overly Simplified Metrics conceal unfair outcomes.
- **OOD Failures:** Models trained on limited data generalize poorly to real-world inputs.

Note: These risks demand proactive design, validation, and monitoring to ensure safe AI deployment.

5.2 Security Concerns

AI systems, similar to conventional software systems, are exposed to dynamic threats and evolving adversarial attacks. However, these challenges are often amplified in AI-based systems due to their reliance on data-driven learning, probabilistic decision-making, and adaptive behaviors, which introduce additional uncertainty and complexity. Over time, these characteristics contribute to increased maintenance burdens, expanded attack surfaces, and greater difficulty in anticipating and mitigating security risks. As a result, these systems are vulnerable to a range of security threats that can introduce technical debt which can in turn introduce other security risks if not mitigated early. This section examines the security concerns associated with identified AI technical debts (AITDs). For each concern, a clear definition is provided, followed by a discussion of the relevant AITDs and their implications. Table 6 presents a comprehensive mapping between these security concerns and their related AITDs, providing an explanatory context for each relationship. It is important to note that many of these security concerns are not exclusive to AI-intensive systems; rather, consistent with our earlier observation that ML systems inherit the maintenance challenges of traditional software while introducing additional AI-specific issues [178], these concerns also arise in conventional systems but are often intensified in AI-enabled contexts due to data dependence, adaptive behavior, and model-driven decision-making.

5.2.1 Authentication and Authorization. This concern arises when AI systems lack robust authentication and authorization mechanisms, which are essential for controlling who can access the system or its data [164]. Without these controls, unauthorized users could gain access to sensitive model outputs or even manipulate system behavior. Inadequate authorization systems can also allow unauthorized users to alter critical system parameters, increasing the risk of security breaches [178]. This is particularly dangerous in safety-critical applications where malicious modifications or unauthorized access could lead to severe consequences, such as in healthcare [206] or autonomous systems [104].

Related AITDs:

- Undeclared Consumers: This AITD is directly related, as it arises from the failure to implement proper access controls and authentication mechanisms, leading to unauthorized usage of the model's outputs.
- Configuration Debt: Misconfigurations can weaken or bypass authentication and authorization mechanisms, increasing security risks.

5.2.2 Data Integrity and Validation. When AI systems do not adequately validate their input data or ensure the integrity of the data they rely on, they become vulnerable to security risks. Data integrity debt occurs when a system begins to process inaccurate, incomplete, or manipulated data, leading to unsafe or unreliable results [68, 164]. For example, in AI systems handling financial transactions, if input data isn't properly validated, attackers might exploit the system by feeding it fraudulent or corrupt data. This could lead to incorrect predictions or actions, such as approving unauthorized transactions or making unsafe decisions based on flawed information.

Related AITDs:

- Unstable Data Dependencies/Data Debt: This directly relates to data integrity issues, as unreliable data sources can introduce incorrect or outdated information into the model, leading to unsafe or insecure outcomes.
- Hidden Feedback Loops: Feedback loops can interfere with the integrity of the system's decision-making process, making it more prone to errors due to invalid data inputs.

5.2.3 Security Patch and Update. This concern arises when security patches and updates for the underlying software, libraries, or dependencies in AI systems are not applied regularly. Over time, unpatched vulnerabilities expose the

system to attacks that exploit known security flaws. Systems relying on outdated or vulnerable libraries are especially prone to exploitation [214]. In safety-critical applications like healthcare or autonomous vehicles, failure to apply security patches can lead to catastrophic outcomes, as attackers might leverage these vulnerabilities to compromise the system's functionality, cause data breaches, or trigger unsafe behaviors.

Related AITDs:

- Dispensable Dependency: Dependencies that are no longer necessary but not removed create vulnerabilities due to missed security patches and updates.
- Inconsistent Use of ML Libraries: Scattered library use makes it difficult to apply security patches consistently across the system, leaving some parts vulnerable to attack.

5.2.4 Data Privacy and Confidentiality. AI systems often handle sensitive data, and failing to implement adequate privacy and confidentiality safeguards creates debt. This includes insufficient encryption, anonymization, or access control for data stored or processed by the system. As a result, unauthorized access to personal or sensitive data can occur, leading to privacy breaches, legal liabilities, and loss of trust [6]. For example, in AI-driven healthcare systems, improper protection of patient records could expose confidential medical information, putting both patients and the organization at risk of legal penalties and reputational damage [144].

Related AITDs:

- Undeclared Consumers: Failure to control access to model outputs can lead to privacy breaches where sensitive data is exposed to unauthorized parties.
- Glue Code: Poorly written integration code may fail to protect data, exposing it to potential leaks or unauthorized access.

5.2.5 Adversarial Attack Resilience. This concern is significant, especially when AI systems are not designed or hardened against adversarial attacks, where input is intentionally manipulated to trick the system into making incorrect or harmful decisions. Attackers may subtly alter input data (such as images or text) in ways that are imperceptible to humans, but cause AI to make erroneous predictions [55]. For instance, an adversarial attack on an autonomous vehicle's AI could involve subtly altering road signs to cause the vehicle to misinterpret a stop sign as a speed limit sign, leading to unsafe driving behaviors [76].

Related AITDs:

- Algorithmic Inclination/Human Bias: AI systems that rely too heavily on specific algorithms may be vulnerable to adversarial attacks that exploit known weaknesses in those algorithms.
- Correction Cascades (CC): Complex dependencies between models may create opportunities for adversarial manipulation, where altering input to one model can cascade through the system, resulting in unexpected and unsafe outcomes.

5.2.6 Fail-Safe Mechanisms. Fail-safe mechanisms are critical for ensuring that AI systems can safely handle unexpected errors, failures, or external threats. In safety-critical environments such as industrial robotics or autonomous driving, the lack of fail-safe mechanisms could result in hazardous conditions when an error occurs, as the system may continue to operate in an unsafe manner instead of shutting down or reverting to a safe state [63].

Related AITDs:

- Dead Experimental Code Paths/Prototype Debt: Experimental or prototype systems often lack robust fail-safe mechanisms, leading to unpredictable behavior when they encounter unexpected situations in production environments.
- Boundary Erosion: As clear separations between components weaken, fail-safe mechanisms may fail to isolate issues, allowing errors in one component to propagate through the system.

5.2.7 Model Interpretability and Transparency. There is concern when AI models are too opaque to allow users to understand how decisions are made, particularly in complex, black-box models like deep neural networks [93, 143]. Without model interpretability, it becomes difficult to detect biases, vulnerabilities, or potential failures, especially in safety-critical applications [93]. For example, in AI-driven medical diagnosis, a lack of transparency in how AI arrives at its recommendations could prevent healthcare professionals from identifying flaws in the model, potentially leading to incorrect treatments or adverse outcomes [113, 136].

Related AITDs:

- Jumbled Model Architecture: Disorganized model architectures can make it difficult to interpret or understand the behavior of the system, increasing the risk of undetected vulnerabilities or unsafe decisions.
- Overly Simplified Metrics: AI systems optimized for abstract performance metrics may lack transparency, making it harder to trace decisions and assess model safety in critical situations.

5.2.8 Real-Time Security Monitoring. AI systems often operate in dynamic environments where real-time threats and security risks must be continuously monitored [46]. An uncertain consequence can arise as a result of insufficient real-time security monitoring and incident response infrastructure in place. Without robust monitoring, security breaches or failures can go unnoticed for long periods, allowing attackers to exploit vulnerabilities or manipulate system behavior undetected [46]. For instance, in AI-controlled energy grids, the absence of real-time security monitoring could allow a cyberattack to disrupt operations without being detected until it is too late.

Related AITDs:

- Hidden Feedback Loops: Without proper real-time monitoring, feedback loops may go unnoticed, leading to system degradation or vulnerabilities that can be exploited over time.
- Dead Experimental Code Paths: Leftover experimental code that is not actively monitored in production can introduce vulnerabilities that are difficult to detect without proper real-time oversight.

5.2.9 Access Control and Data Protection. A lack of sufficient controls over who can access sensitive data or model outputs in AI systems is a significant concern in AI systems [207]. If access controls are weak or nonexistent, unauthorized users could gain access to data that should be protected, such as proprietary model outputs or sensitive customer data. This increases the risk of data breaches and compliance violations, particularly in highly regulated industries such as finance [91] or healthcare [59]. Furthermore, in safety-critical systems, unauthorized access to model outputs could lead to unsafe actions if the model predictions are used inappropriately [170].

Related AITDs:

- Undeclared Consumers: As this AITD involves lack of control over who accesses model outputs, it directly contributes to data protection issues, making it difficult to secure sensitive information.
- Deep God File: A large, monolithic file containing multiple components makes it harder to enforce access controls effectively, increasing the risk of data leaks.

5.2.10 Complexity-Induced Vulnerabilities. AI systems with complex architectures or pipelines often hide vulnerabilities that are difficult to detect and address [178]. This occurs when the system's complexity makes it challenging to perform thorough security assessments or safety analyses. For example, in systems where multiple models or components are interdependent, it may be difficult to track how data flows through the system or identify weak points where security measures are inadequate. This complexity can lead to vulnerabilities that attackers can exploit or cause system failures in safety-critical applications.

Related AITDs:

- Pipeline Jungle: The complexity of AI pipelines increases the likelihood of hidden vulnerabilities, making it harder to secure the system and identify potential weaknesses.
- Entanglement: Tightly coupled components make it difficult to manage security risks, as vulnerabilities in one part of the system can affect other interconnected components.

5.2.11 Dependency-Related Vulnerabilities. AI systems often rely on external libraries, tools, and frameworks to function. Whenever outdated or unused dependencies are not properly managed, it leads to creating vulnerabilities that can be exploited by attackers [42]. Unused dependencies that remain in the system without regular updates can serve as backdoor for malicious activity [51]. In safety-critical applications such as autonomous driving or healthcare, dependency-related vulnerabilities can lead to unpredictable system behavior or integration issues, potentially causing unsafe outcomes or system failures.

Related AITDs:

- Dispensable Dependency: Unnecessary dependencies in the system introduce security vulnerabilities, as they are often not updated or patched regularly.
- Compatibility Debt: Dependency on outdated or insecure libraries can result in vulnerabilities that compromise system security and safety.

5.2.12 Ethical and Bias-Induced Safety and Security. AI systems that are developed without adequate consideration for ethical principles, such as fairness, accountability, and transparency, accumulate ethical debt [158] can lead to biased decisions that result in unsafe or unethical outcomes [168]. In safety-critical environments, such as healthcare, biased AI models could deny certain demographic groups access to critical treatments, resulting in harm [201]. Similarly, from a security perspective, attackers could exploit biased algorithms to influence the system's behavior in ways that compromise both safety and security [42].

Related AITDs:

- Ethics Debt: This is directly related, as biases in AI models can lead to unethical decisions, which may also introduce security risks if attackers exploit those biases.
- Algorithmic Inclination/Human Bias: Over-reliance on specific algorithms may introduce systemic bias, leading to unsafe decisions in critical applications and increasing susceptibility to exploitation.

These security concerns are deeply interwoven with technical debt manifestations across AI-enabled systems. Figure 5 visually consolidates these relationships, highlighting their interconnections, providing a graphical representation of how each AITD contributes to broader risk categories.

RQ2.2 – Security Concerns in AI Systems

The identified AITDs, such as Undeclared Consumers, Data Debt, Algorithmic Bias, and Ethical Debt, significantly impact the security of AI-based systems. These debts can lead to vulnerabilities, including unauthorized access, data breaches, and biased outcomes. Critical risks identified include:

- Authentication and Authorization: Weak controls can lead to unauthorized access to sensitive AI model outputs.
- Data Integrity: Poor validation processes make systems vulnerable to manipulated or inaccurate data.
- Adversarial Attacks: Systems without resilience mechanisms are prone to exploitation by adversarial inputs, jeopardizing both security and safety.
- Ethical and Bias-Induced: Failing to address fairness and accountability in AI systems can lead to harmful or unethical outcomes.

Impact on High-Stakes Domains: The safety-critical nature of sectors like healthcare, finance, and autonomous systems magnifies these concerns, requiring robust mitigation strategies.

6 Safety and Security guidelines for mitigating the identified AITDs (RQ3)

6.1 Safety Guidelines

To mitigate the impact of AI Technical Debts (AITDs) on system safety, this study identifies a set of strategic guidelines derived from established AI safety research and best practices. These guidelines address core safety concerns, including robustness, explainability, fairness, transparency, and operational assurance. Each guideline is explicitly linked to specific AITDs that introduce safety risks, as identified in Section 4. Our contribution lies in systematically consolidating, synthesizing, and mapping these established guidelines to the concrete forms of AITD uncovered in this review.

6.1.1 Explainable AI (XAI). Explainable AI encompasses methods and techniques that make AI system decisions understandable to humans. This includes model-agnostic tools like Local Interpretable Model-agnostic Explanations (LIME) and SHapley Additive exPlanations (SHAP), inherently interpretable models like decision trees, and visualizations like saliency maps. By enhancing transparency, XAI helps end-users, developers, and regulators to audit and trust AI outputs [2, 8, 39, 58, 111, 183, 221].

Related AITDs: *Jumbled Model Architecture* obscures the internal structure and decision pathways of models, complicating explanation. *Overly Simplified Metrics* do not convey interpretable performance indicators. *Documentation Debt* limits external understanding of design choices and model logic.

6.1.2 Fairness Assessment and Bias Mitigation. This guideline ensures that AI systems treat individuals and groups equitably. Techniques include fairness-aware training algorithms [185], pre-processing data balancing [152], and post-hoc fairness evaluations using metrics like demographic parity or equal opportunity [89]. The goal is to detect and reduce both data and algorithmic biases [66, 89].

Related AITDs: *Algorithmic Inclination / Human Bias* encodes unchecked human biases into model logic. *Ethical Debt* stems from ignoring fairness and social implications during development. *Overly Simplified Metrics* may fail to capture nuanced biases in model outputs.

Table 6. Mapping Security Concerns to Related AI Technical Debts (AITDs)

S/N	Security Concern	Related AITDs	Explanation
1	Authentication and Authorization	Undeclared Consumers, Configuration Debt, Glue Code	Lack of access control and misconfigurations allow unauthorized access to model outputs and parameters.
2	Data Integrity and Validation	Data Debt (Unstable Dependencies), Hidden Feedback Loops, Correction Cascades	Inadequate validation of input data or changes in source data can compromise prediction accuracy and trustworthiness.
3	Security Patch and Update	Dispensable Dependency, Scattered Use of ML Libraries, Compatibility Debt, Multiple Language Smells	Outdated, scattered, or heterogeneous codebases increase difficulty in applying consistent security updates.
4	Data Privacy and Confidentiality	Undeclared Consumers, Glue Code, Deep God File	Poor control of model outputs and monolithic code structures may expose sensitive data.
5	Adversarial Attack Resilience	Algorithmic Inclination / Human Bias, Correction Cascades, Entanglement	Lack of algorithm diversity and complex model interdependencies increase susceptibility to adversarial manipulation.
6	Fail-Safe Mechanisms	Prototype Debt, Boundary Erosion, Jumbled Model Architecture	Weak architectural modularity and experimental code can prevent safe fallback during system failure.
7	Model Interpretability and Transparency	Jumbled Model Architecture, Overly Simplified Metrics	Opaque model design and inappropriate evaluation metrics hinder understanding and safety validation.
8	Real-Time Security Monitoring	Hidden Feedback Loops, Prototype Debt	Insufficient oversight allows critical issues (e.g., feedback drift) to go unnoticed in production..
9	Access Control and Data Protection	Undeclared Consumers, Glue Code, Deep God File	Lack of modularity and loosely managed outputs can expose the system to unauthorized use or data leakage.
10	Complexity-Induced Vulnerabilities	Pipeline Jungle, Entanglement, Configuration Debt, Multiple Language Smells	Interdependent modules and use of multiple languages increase brittleness and vulnerability to subtle configuration flaws.
11	Dependency-Related Vulnerabilities	Dispensable Dependency, Compatibility Debt	Poor management of software/library dependencies increases exposure to known exploits.
12	Ethical and Bias-Induced Safety	Ethical Debt, Algorithmic Inclination / Bias, Overly Simplified Metrics	Neglecting fairness, transparency, and appropriate evaluation criteria can lead to unsafe, discriminatory outcomes.

6.1.3 Adversarial Training. Adversarial training improves robustness by exposing models to adversarial examples during training. This enables the system to learn resilience against malicious perturbations. It is crucial for security-critical applications where robustness is a primary safety concern [226, 231].

Related AITDs: *Algorithmic Inclination / Human Bias* may prioritize performance over resilience. *Correction Cascades* allow adversarial perturbations in one module to affect downstream outputs. *Entanglement* creates interdependencies that amplify security vulnerabilities.

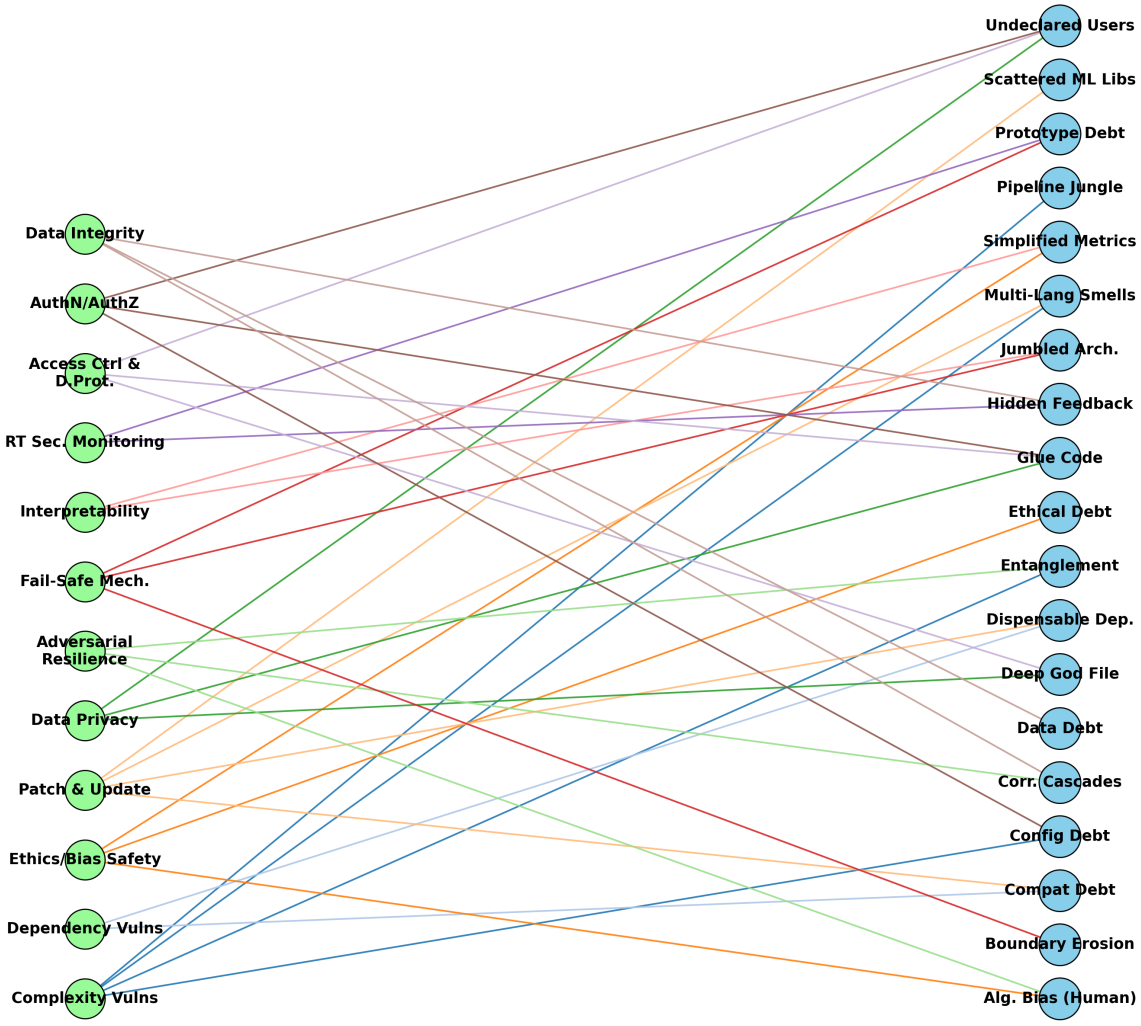

Fig. 5. Bipartite mapping between security concerns and AI Technical Debts (AITDs), showing how identified AITDs contribute to or exacerbate specific security risks in AI-enabled systems. Green circles represent security concerns, while blue circles represent AITDs.

6.1.4 Out-of-Distribution (OOD) Detection. OOD detection allows AI systems to recognize when they are presented with inputs that significantly deviate from their training data. Strategies include confidence scoring, uncertainty modeling, and Bayesian inference. By deferring or flagging these cases, the system avoids unsafe decisions [87, 103].

Related AITDs: *Pipeline Jungle* complicates systematic integration of OOD detectors. *Prototype Debt* results in fragile components not validated for distributional robustness. *Overly Simplified Metrics* often miss edge-case or anomaly behavior.

6.1.5 Formal Verification. Formal verification applies mathematical proofs and logical reasoning to ensure that AI systems meet predefined specifications. This is particularly useful for verifying safety properties, such as invariants and boundary conditions in control systems [50, 134].

Table 7. Mapping of Safety Guidelines to Related AI Technical Debts (AITDs)

S/N	Safety Guideline	Related AITDs	Explanation
1	Explainable AI (XAI)	Jumbled Model Architecture, Overly Simplified Metrics, Documentation Debt	Improves model transparency and enables users to understand and justify AI decisions, enhancing trust and accountability.
2	Fairness Assessment and Bias Mitigation	Algorithmic Inclination / Human Bias, Ethical Debt, Overly Simplified Metrics	Detects and corrects biases in data and model behavior to ensure equitable and non-discriminatory AI outcomes.
3	Adversarial Training	Algorithmic Inclination / Human Bias, Correction Cascades, Entanglement	Increases model robustness against malicious inputs and reduces vulnerability propagation through tightly coupled components.
4	OOD Detection	Pipeline Jungle, Prototype Debt, Overly Simplified Metrics	Prevents unsafe decisions on unfamiliar data by identifying distribution shifts and handling anomalies effectively.
5	Formal Verification	Defect Debt, Test Debt, Jumbled Model Architecture	Ensures correctness of AI systems via mathematical proofs, addressing unverified bugs and untestable complex models.
6	Domain Adaptation and Transfer Learning	Data Debt, Pipeline Jungle, Dead Experimental Code Paths	Enhances model generalization across diverse contexts by adapting to new domains and discarding obsolete or fragile components.
7	Black Box Auditing and Logging	Documentation Debt, Versioning Debt	Facilitates traceability and compliance through systematic logging and monitoring of AI decision-making processes.
8	Secure and Federated Learning	Configuration Debt, Dispensable Dependency, Compatibility Debt	Preserves data privacy and system integrity by securely distributing training without centralizing sensitive information.

Related AITDs: *Defect Debt* includes unaddressed errors that may invalidate verification results. *Test Debt* results in insufficient test coverage for verifying formal properties. *Jumbled Model Architecture* complicates reasoning due to its opacity and lack of modularity.

6.1.6 Domain Adaptation and Transfer Learning. These approaches allow models trained on one domain to adapt effectively to another. They involve fine-tuning or augmenting training data to generalize across varying distributions [102, 230]. These are essential for deploying models in dynamic or diverse environments.

Related AITDs: *Data Debt* limits generalization due to insufficient diversity. *Pipeline Jungle* obstructs the integration of domain adaptation pipelines. *Dead Experimental Code Paths* leave legacy components that are poorly adapted to new domains.

6.1.7 Black Box Auditing and Logging. This involves systematically recording the inputs, outputs, and internal states of AI systems to enable retrospective analysis [2]. Logging enables debugging, monitoring, and compliance with regulatory standards. It is essential for safety transparency [45].

Related AITDs: *Documentation Debt* restricts the reproducibility of logs and audit trails. *Versioning Debt* impairs traceability across system updates.

6.1.8 Secure and Federated Learning. These are privacy-preserving learning paradigms where models are trained collaboratively without sharing raw data [225]. Federated learning improves security and trust in multi-agent environments. Encryption and differential privacy techniques are often employed [184].

Related AITDs: *Configuration Debt* creates exposure due to insecure settings. *Dispensable Dependency* brings in third-party components that may not be vetted for privacy. *Compatibility Debt* hinders integration of federated architectures into existing systems.

These guidelines provide a multidimensional defense-in-depth strategy that enhances the safety of AI-based systems while also facilitating sustainable debt management. As summarized in Table 7, each safety guideline is explicitly linked to one or more AI Technical Debts (AITDs), highlighting the targeted impact of these practices. Additionally, Figure 6 provides a visual mapping that illustrates how each guideline contributes to the mitigation of specific AITDs, supporting a comprehensive safety assurance strategy.

RQ3.1: Safety Guidelines for Mitigating AITDs

To enhance the safety of AI-based systems, the following guidelines are recommended for addressing AI Technical Debts (AITDs):

- **Explainable AI (XAI):** Enhances system transparency, mitigating debts related to interpretability and trust.
- **Robustness Testing:** Detects failure points under adversarial or noisy conditions.
- **Continuous Monitoring:** Ensures early detection of drift, decay, or bias during deployment.
- **Ethical and Fairness Audits:** Mitigates bias, ethical, and societal debts through regular assessments.
- **Modular and Versioned Pipelines:** Reduces architecture and documentation debt by enabling traceability and maintainability.
- **OOD Detection Mechanisms:** Handles novel inputs and prevents unsafe behavior from unrecognized data.

Adopting these guidelines supports resilient, fair, and transparent AI systems in safety-critical domains.

6.2 Security Guidelines

This section presents a comprehensive set of security guidelines designed to effectively mitigate the AI Technical Debts (AITDs) identified in this review. Our contribution lies in systematically consolidating, synthesizing, and mapping established guidelines to the specific forms of AITD uncovered in this study. The selected guidelines reflect the most relevant and impactful practices for addressing the root causes of AITDs, spanning critical areas such as secure data handling, adversarial resilience, configuration discipline, system monitoring, dependency governance, and organizational preparedness. In the following subsections, each guideline is presented with: (i) a concise definition, (ii) a practical example illustrating its application in real-world AI deployments, and (iii) an explanation of the specific AITDs it helps mitigate. To enhance traceability and support evidence-based mitigation planning, Table 8 provides a structured mapping between the security guidelines and the AITDs they address, while Figure 7 visually illustrates their interconnections. Together, these artifacts enable practitioners and researchers to understand how targeted security practices can reduce the accumulation of technical debt across the AI lifecycle and strengthen the robustness, trustworthiness, and sustainability of AI-enabled systems.

6.2.1 Configuration Management. Robust configuration management ensures that all AI components—models, pipelines, hyperparameters, dependencies, and deployment environments—are consistently versioned, controlled, and auditable throughout the lifecycle [96, 191, 196]. In AI-enabled systems, where small changes in data sources, hyperparameters, or library versions can significantly alter model behavior, disciplined configuration practices are essential to maintain reproducibility, traceability, and secure rollback in case of failures or regressions. Centralizing configuration artifacts (e.g., as code, in registries, or configuration databases) also supports governance, compliance, and cross-team collaboration in complex MLOps settings.

- Example: An MLOps pipeline stores configuration snapshots (hyperparameters, model versions, data schemas) in a managed registry before every retraining cycle, ensuring reproducibility.
- Related AITDs: Configuration Debt (uncontrolled parameter drift), Versioning Debt (untracked changes across updates), Pipeline Jungle (inconsistent configurations across stages), Compatibility Debt (misaligned module configurations).

6.2.2 Encryption and Secure Data Storage. Encryption and secure data storage are foundational security practices for AI systems, ensuring that all sensitive data—during collection, preprocessing, model training, evaluation, and deployment—is protected from unauthorized access, tampering, or interception. This guideline encompasses two complementary aspects: encryption of data at rest and in transit and the secure storage of training, validation, and retraining datasets, especially those containing personally identifiable or confidential information. Compliance-focused AI systems must implement industry-grade cryptographic standards (e.g., AES-256 for stored data, TLS 1.2+ for data in transit), robust key-management procedures, and controlled access to secure storage environments. These safeguards align with regulatory requirements such as GDPR and ISO 27002 by ensuring confidentiality, integrity, and long-term protection of sensitive data throughout the AI lifecycle [88, 92, 176, 180, 202].

- Example: In a healthcare diagnostic AI system, patient records uploaded for model inference are transmitted over encrypted channels (TLS), while historical datasets used for retraining are stored in encrypted repositories (AES-256) with strict access controls. This prevents unauthorized access, accidental exposure, or inference-time data leakage.
- Related AITDs: This guideline mitigates several data- and configuration-related debts, including: *Data Debt / Unstable Data Dependencies* — securing data flows reduces exposure to compromised or inconsistent data sources. *Configuration Debt* — enforcing proper cryptographic settings, key rotation, and secure storage parameters prevents misconfigurations that weaken system security. *Documentation Debt* — requiring well-documented encryption policies and data-handling procedures enhances traceability and audit readiness. *Versioning Debt* — secure key management and versioning of encrypted assets prevent stale or misaligned cryptographic states.

6.2.3 Continuous Behavioral & Performance Monitoring. This involves the systematic, real-time observation of AI system behavior to ensure that models operate within expected ethical, functional, and performance boundaries. This unified guideline integrates two complementary monitoring practices: (i) continuous behavioral monitoring, which evaluates adherence to ethical, fairness, and security policies; and (ii) performance monitoring, which tracks accuracy, drift, degradation, and operational reliability over time. Together, these monitoring practices enable early detection of anomalies, misconfigurations, emerging biases, model drift, or hidden system interactions that may compromise system integrity or safety [30, 65, 81, 88, 96, 126]. Continuous monitoring is especially critical for AI systems deployed

in dynamic or high-stakes environments where data distributions evolve and decision quality must remain consistently reliable.

- Example: In a real-time fraud detection system, continuous monitoring evaluates whether model predictions remain fair and unbiased across customer groups, while performance monitoring checks for drops in detection accuracy that may indicate emerging fraud patterns or model drift. Alerts are triggered when behavioral or performance deviations exceed thresholds, prompting retraining, recalibration, or human review.
- Related AITDs: This guideline mitigates multiple AITDs, including: *Ethical Debt* — by continuously checking for fairness, bias, or ethical violations in model outputs; *Configuration Debt* — by detecting misconfigurations or runtime deviations that affect behavior or safety; *Unstable Data Dependencies (Data Debt)* — by identifying drift or changes in upstream data that degrade model performance; *Hidden Feedback Loops* — by surfacing recurrent or cyclic interactions that cause unexpected model behaviors; and *Performance or Design Debt* — by detecting accuracy drops, latency spikes, or reliability issues early in the AI lifecycle.

6.2.4 Data Validation. This guideline requires that all input data feeding AI pipelines undergo strict validation checks for correctness, format consistency, completeness, and authenticity [88, 163, 182]. Strong validation reduces the risk of corrupted, noisy, or malicious data introducing instability.

- Example: Before retraining a recommendation system, automated validators check for missing fields, incorrect data types, and out-of-distribution samples, preventing corrupted data from degrading model accuracy.
- Related AITDs: Data Debt (unvetted data introduces drift and errors), Pipeline Jungle (complex pipelines lacking validation checkpoints), Correction Cascades (invalid data propagating errors downstream).

6.2.5 Secure Coding Practices. This guideline emphasizes applying standard secure-software engineering principles—such as static analysis, dependency scanning, code reviews, and avoiding hard-coded secrets—to AI pipelines and model-serving infrastructure [96, 122, 191].

- Example: During deployment, static code analysis flags insecure API endpoints and outdated dependencies used in the inference server, prompting corrective refactoring.
- Related AITDs: Glue Code Debt (quick fixes introducing vulnerabilities), Duplicate Model Code (security flaws replicated across modules), Unwanted Debugging Code (leftover logs leaking sensitive info), Defect Debt (known issues not addressed).

6.2.6 Incident Response for AI Systems. Incident response involves establishing structured procedures to detect, investigate, contain, and recover from security incidents affecting AI workflows [88, 99, 101, 132]. This includes AI-specific events such as adversarial data poisoning, unauthorized model access, or misconfiguration-induced failures.

- Example: In a fraud-detection AI system, if an attacker injects adversarial inputs that manipulate prediction thresholds, an incident response workflow automatically detects anomalies, isolates compromised components, and triggers human review.
- Related AITDs: Hidden Feedback Loops (undetected abnormal system behavior), Unstable Data Dependencies (data poisoning events), Configuration Debt (incidents caused by misconfigurations).

6.2.7 Threat Intelligence. It involves the continuous collection, analysis, and application of information about emerging vulnerabilities, attack vectors, system misuses, and adversarial behaviors relevant to AI-enabled systems. In the context of AI security, threat intelligence extends beyond traditional cybersecurity practices by incorporating AI-specific risks such

as data poisoning, model evasion, adversarial perturbations, prompt injection, and manipulation of training or inference pipelines. By proactively monitoring threat landscapes—including AI model repositories, dependency ecosystems, upstream data sources, and adversarial research communities—organizations can anticipate misconfigurations, model weaknesses, and infrastructural exposures that could manifest as AI Technical Debt (AITD). Integrating actionable threat intelligence into AI development pipelines enhances preparedness, supports risk-informed decision-making, and ensures that AI components evolve in alignment with emerging security requirements [36, 96, 174].

- Example: A financial-services AI platform incorporates threat intelligence feeds that track newly discovered vulnerabilities in ML libraries (e.g., TensorFlow, PyTorch), recent data poisoning campaigns detected in public datasets, and adversarial evasion techniques emerging from academic research. When a critical vulnerability is reported in a dependency, the system automatically flags relevant components, triggering audits, dependency upgrades, and revalidation of affected models.
- Related AITDs: Threat Intelligence mitigates a range of debt types, including: *Dispensable Dependency* — by detecting obsolete or vulnerable dependencies that should be removed or replaced; *Scattered Use of ML Libraries (SML)* — by identifying fragmented or inconsistent library usage that increases exposure to security flaws; *Configuration Debt* — by uncovering unsafe or outdated configuration settings tied to evolving threat patterns; *Data Debt / Unstable Data Dependencies* — by detecting compromised or manipulated upstream data sources; *Boundary Erosion and Compatibility Debt* — by highlighting insecure integrations or outdated interfaces that may become attack vectors; *Documentation and Versioning Debt* — by reinforcing the need for traceability of updates, patches, and threat responses across AI lifecycle artifacts.

6.2.8 Red Teaming and AI Alignment (RLHF). Red teaming systematically tests AI systems using adversarial, stress-testing, or probing techniques to expose vulnerabilities. RLHF (Reinforcement Learning from Human Feedback) aligns model behavior with safe and acceptable outcomes, reducing misuse and unintended behaviors [22, 64, 150, 197, 211].

- Example: Before deployment, a large language model undergoes red-teaming sessions where experts test harmful prompts, manipulations, and jailbreak attempts. RLHF is then used to retrain the model and correct unsafe behaviors.
- Related AITDs: Ethical Debt (lack of alignment with societal norms), Algorithmic Inclination / Bias (unchecked bias requiring feedback alignment), Overly Simplified Metrics (models optimized with insufficient safety metrics).

6.2.9 Data Leak Prevention (DLP). Data Leak Prevention involves monitoring, filtering, and controlling the flow of sensitive data inside AI pipelines to prevent accidental exposure or exfiltration [13, 96, 97, 191].

- Example: A model-training cluster blocks outbound file transfers containing sensitive patterns (e.g., credit card numbers), preventing unintended dataset exposure.
- Related AITDs: Data Debt (poorly governed sensitive data), Documentation Debt (missing data-handling guidelines), Dispensable Dependency (unnecessary third-party components leaking data), Undeclared Consumers (unmonitored data usage).

6.2.10 Data Masking & Information Deletion. Data masking anonymizes sensitive data used in training or inference. Information deletion ensures obsolete or risky data is securely removed, minimizing retention-related risk [29, 72, 96, 191].

- Example: An AI system masks personally identifiable information (PII) before performing analytics and automatically deletes expired training logs after a retention window.
- Related AITDs: Data Debt (unmasked or outdated data remains in the pipeline), Unstable Data Dependencies (stale data causing unexpected behavior), Process/Infrastructure Debt (missing secure-deletion procedures), Documentation Debt (unclear retention policies).

6.2.11 Controlled AI Output Consumers. This guideline ensures that outputs produced by AI models—predictions, scores, embeddings, or generated content—are only accessed by explicitly authorized systems, services, or users [30, 130, 164]. Enforcing strict consumption policies prevents downstream components from silently depending on AI outputs, which can introduce hidden couplings, regulatory violations, and unmonitored propagation of errors or sensitive information. Controlled output consumption also supports auditability by requiring explicit logging, authentication, and authorization checks whenever AI outputs are requested or consumed. This reduces the risk of unintended reuse, shadow pipelines, or unauthorized integrations that frequently lead to systemic vulnerabilities in AI-driven architectures.

- Example: In an AI-driven cybersecurity monitoring platform, only vetted detection modules and SOC (Security Operations Center) tools can access threat-classification outputs through authenticated API calls. Unauthorized services attempting to query the AI's outputs are automatically blocked and logged, preventing silent dependencies or accidental leakage of sensitive threat intelligence.
- Related AITDs: *Undeclared Consumers* — prevents hidden or unauthorized modules from silently consuming AI outputs; *Glue Code Debt* — reduces ad-hoc integrations that bypass approved consumption pathways; *Documentation Debt* — requires clear documentation of permitted consumers and access rules.

6.2.12 Model Watermarking (Model Integrity Controls). Model watermarking embeds identifiers or verification signatures into AI models to detect unauthorized replication, tampering, or misuse [23, 133, 145, 160]. This strengthens model integrity and provides forensic traceability.

- Example: A commercial NLP model embeds a proprietary watermark into its weight matrices. If leaked or cloned, developers can verify fingerprints to identify unauthorized deployments.
- Related AITDs: Undeclared Consumers (untracked or unauthorized model usage), Versioning Debt (lack of traceability enabling misuse), Documentation Debt (missing model-ownership records), Glue Code Debt (uncontrolled integrations facilitating illicit replication).

6.2.13 AI System Software Bill of Materials (AI-SBOM). An AI System Software Bill of Materials (AI-SBOM) is a machine-readable inventory that lists all datasets, model versions, libraries, configurations, and dependencies used across the AI lifecycle [161, 195, 218]. It provides end-to-end traceability of how models were trained, what data they relied on, and which components they depend on, thereby strengthening transparency, auditability, and supply-chain security. AI-SBOMs also support rapid vulnerability assessment when flaws or dataset integrity issues arise [219].

- Example: In a healthcare AI system, an AI-SBOM records training datasets, preprocessing steps, model checkpoints, and library versions, enabling auditors to verify compliance with GDPR and medical-device regulations.
- Related AITDs: *Data Debt / Unstable Data Dependencies* — through clear provenance tracking. *Dispensable Dependency* — by identifying outdated or unnecessary components. *Versioning Debt* — by documenting model and dependency histories. *Documentation Debt* — by formalizing system lineage and configuration details.

6.2.14 Modular & Reusable AI Components. This guideline emphasizes designing AI systems using modular, self-contained components with clearly defined interfaces and separation of concerns [38, 131, 167, 176]. Modular architectures minimize unnecessary coupling between data pipelines, model logic, and service layers, making AI systems easier to test, secure, maintain, and evolve. Reusable components also promote architectural consistency across projects, reduce duplication, and lower the risk of integrating ad-hoc patches or experimental code that later becomes technical debt.

- Example: In an AI-based fraud detection platform, independently developed modules for feature extraction, model inference, and case scoring can be reused across credit card fraud, loan fraud, and identity-theft scenarios. This avoids rewriting redundant logic and enables secure updates without impacting unrelated components.
- Related AITDs: *Glue Code Debt* — modular interfaces reduce the need for brittle integration scripts. *Deep God File* — decomposition prevents monolithic files accumulating multiple responsibilities. *Design Debt* — modularity enforces cleaner architectural boundaries. *Duplicate Model Code* — reusable components reduce redundant implementations across the system.

6.2.15 Standardized & Interoperable Data Formats. This guideline promotes the use of consistent, well-defined, and interoperable data formats across all stages of the AI pipeline [30, 80, 141, 229]. Standardization minimizes the need for ad-hoc data conversions and reduces the likelihood of mismatches between feature schemas, model inputs, and downstream services. By enforcing shared data conventions—such as unified schemas, typed data structures, consistent encodings, and versioned dataset specifications—organizations can prevent integration failures, enhance security by reducing transformation-related vulnerabilities, and improve long-term maintainability of AI workflows.

- Example: In a smart city AI platform aggregating data from traffic sensors, public transit feeds, and environmental monitors, enforcing interoperable formats (e.g., standardized timestamps, unified geospatial encodings, consistent JSON schemas) ensures that downstream prediction models can reliably process data without error-prone conversion scripts.
- Related AITDs: *Glue Code Debt* — reduces the need for excessive data wrangling or conversion scripts used to bridge incompatible formats; *Unstable Data Dependencies (Data Debt)* — ensures predictable, consistent data flows across heterogeneous systems; *Compatibility Debt* — avoids failures originating from mismatched data structures between models, services, or pipeline components.

6.2.16 API-Governed AI Access Control. This guideline promotes accessing AI models and components exclusively through well-governed, authenticated, and rate-limited APIs rather than distributing models for local execution [88, 126, 151, 192]. Centralizing access through APIs strengthens security by enforcing fine-grained permissions, monitoring usage patterns, applying request validation, and preventing unauthorized or uncontrolled consumption of AI outputs. It also reduces the attack surface by avoiding model proliferation across uncontrolled environments, enabling organizations to maintain better visibility, governance, and lifecycle control over deployed AI capabilities. In addition, API-mediated access supports compliance and auditability, since all interactions with the AI system can be logged, attributed, and reviewed, ensuring that AI predictions and model updates remain traceable and accountable.

- Example: An AI-based weather prediction model is deployed behind a secure API gateway. External partners in agriculture, aviation, and logistics can query predictions through authenticated API calls, without receiving direct access to model weights or internal logic. This prevents model theft, unauthorized reuse, and uncontrolled distribution, while enabling centralized monitoring and permission enforcement.

- Related AITDs: *Undeclared Consumers* — by ensuring AI outputs are accessed only by authorized systems and recorded in access logs. *Dispensable Dependency* — by reducing the need to embed or replicate models locally, avoiding unnecessary third-party integrations that create security and maintenance burdens. *Glue Code Debt* — by minimizing ad-hoc local integrations and promoting standardized, well-defined API interfaces. *Versioning Debt* — by enabling centralized version control of models and ensuring clients always access the correct model version through the API.

6.2.17 Adequate AI Resource & Continuity Management. This focuses on ensuring that AI systems are provisioned with adequate computational resources—such as CPU/GPU capacity, memory, storage, and network bandwidth—while also maintaining organizational readiness for operational continuity in the event of outages, failures, or unexpected load surges. This guideline combines two complementary security and reliability practices: (i) resource sufficiency, which supports stable and timely AI model execution, and (ii) ICT readiness for business continuity, which ensures fallback mechanisms, redundancy, and systematic recovery procedures are in place to maintain secure and safe AI operation even under adverse conditions [30, 96, 180, 187].

- Example: In an autonomous driving system, ensuring sufficient GPU acceleration and memory bandwidth allows AI models to process multimodal sensor inputs (e.g., LiDAR, radar, cameras) with millisecond latency. Simultaneously, redundancy in compute nodes and a business continuity plan (e.g., automatic failover to a secondary control module) ensures that vehicle perception and decision-making continue uninterrupted during hardware faults or peak load conditions.
- Related AITDs: This guideline mitigates several AITDs, including: *Configuration Debt* — by ensuring proper resource provisioning and avoiding misconfigured system parameters that lead to performance bottlenecks; and *Process/Infrastructure Debt* — by strengthening operational readiness, redundancy, and failover strategies.

6.2.18 AI Component Divergence Monitoring. This guideline focuses on the continuous detection and analysis of behavioral discrepancies between redundant AI components or between an AI system's expected outputs and its actual runtime behavior [1, 126, 166]. Such divergence may signal a variety of underlying issues—including data quality problems, configuration drift, adversarial interference, model degradation, or faulty integration logic. By systematically monitoring for mismatches, organizations can identify anomalies at an early stage, initiate corrective action, and prevent security- or safety-critical consequences. Divergence monitoring is especially important in safety-critical and real-time AI systems, where silent failures or unnoticed drift can propagate downstream and trigger cascading errors.

- Example: In an AI-driven industrial control system, if two redundant models controlling a robotic arm produce different movement instructions, this discrepancy would trigger an alert for human review. The system may automatically revert to a safe fallback mode or require operator approval before executing actions, thereby preventing accidental equipment damage or unsafe operations.
- Related AITDs: Helps mitigate *Unstable Data Dependencies* by catching inconsistencies in upstream data or model behavior early, and *Correction Cascades* by preventing erroneous or misaligned outputs from propagating through interconnected components in the AI pipeline.

6.2.19 Multi-Model Consensus Decision Framework. This guideline advocates deploying multiple AI models in parallel—each trained with different architectures, datasets, feature sets, or optimization strategies—to independently generate decisions that are then compared to derive a consensus [82, 126, 176, 179]. By diversifying the decision-making process across multiple models, the system becomes inherently more fault-tolerant, reducing susceptibility

to single-model failures, dataset biases, adversarial perturbations, or unexpected distribution shifts. Consensus-based decision mechanisms strengthen the robustness of AI-driven systems by identifying anomalous outputs early, isolating underperforming models, and enabling ensemble-level validation [56]. Moreover, multi-model consensus supports adaptive system design, allowing continuous improvement by monitoring divergence patterns across models and informing when retraining, recalibration, or model retirement is necessary.

- Example: In healthcare diagnostics, separate AI models—such as a CNN, a transformer-based classifier, and a segmentation-based model—can independently analyze radiographic images. If one model generates an output that deviates from the majority consensus, the system automatically flags the case for manual review, ensuring higher reliability and preventing diagnostic errors.
- Related AITDs: Mitigates *Correction Cascades* by preventing downstream components from acting on a faulty single-model output. Addresses *Hidden Feedback Loops* by distributing decision-making across diverse model pathways, reducing the risk that self-reinforcing errors or biased predictions propagate through the pipeline without detection.

6.2.20 AI Redundancy & Failover Mechanisms. This guideline emphasizes the deployment of multiple identical—or functionally equivalent—AI components operating in parallel to provide resilience, robustness, and continuity under failure conditions [83, 139, 209]. By maintaining redundant AI modules, the system can seamlessly transition to a backup component whenever the primary model encounters unexpected behavior, hardware degradation, adversarial interference, or operational faults. Failover mechanisms are essential in high-availability and safety-critical environments, where a single point of AI failure can produce cascading errors or compromise system safety. Redundancy reduces operational risk by ensuring that model predictions remain stable even when one instance becomes compromised, overloaded, or misconfigured. It also enables cross-validation between identical components, improving anomaly detection and strengthening reliability during runtime. Beyond resilience, redundancy supports secure and responsible AI deployment by enabling rolling updates, safe model rollbacks, and phased testing of new model versions. This ensures that experimental or updated AI components do not destabilize system behavior, as stable redundant units remain available to maintain uninterrupted operation. Collectively, redundancy and failover strategies enhance system robustness, protect against adversarial failure modes, and uphold service continuity.

- Example: In a smart grid energy distribution system, multiple redundant AI controllers continuously monitor load patterns and grid conditions. If the primary controller is compromised—due to a cyberattack, hardware fault, or sensor malfunction—the secondary controller instantly assumes control, preventing blackouts, safety hazards, or cascading failures across the grid.
- Related AITDs: Mitigates *Algorithmic Inclination / Human Bias* by reducing dependence on a single algorithmic decision-maker. Addresses *Unstable Data Dependencies* by ensuring that failures in one data pathway do not compromise the entire system, as redundant components can operate on parallel or validated data streams.

6.2.21 Use of Hardened Models and Defensive Training. This guideline advocates for using models trained with defensive mechanisms such as adversarial training, robust optimization, or certified defenses to increase resilience against malicious perturbations [88, 149, 173].

- Example: An image classifier used in a biometric access system is trained with adversarially perturbed samples, increasing robustness against evasion attacks that slightly manipulate facial images.

- Related AITDs: Algorithmic Inclination / Human Bias (models optimized only for accuracy, not robustness), Entanglement (high interdependencies amplifying vulnerabilities), Correction Cascades (unrobust models propagating errors), Design Debt (architectures not built for robust defense).

6.2.22 Input Sanitization (Defense Against Evasion Attacks). This involves applying filters, normalization techniques, and anomaly detection to remove adversarial perturbations or malicious payloads [24, 88, 108]. This is critical for mitigating evasion attacks targeting model vulnerabilities.

- Example: A vision-based authentication model normalizes pixel distributions and rejects suspiciously manipulated images before inference, reducing exposure to adversarial examples designed to bypass access controls.
- Related AITDs (not sure - double check): Algorithmic Inclination / Human Bias (models not trained to handle edge cases), Entanglement (tight coupling amplifying adversarial effects), Correction Cascades (malicious inputs causing downstream failures), Defect Debt (unhandled edge cases left unresolved).

6.2.23 Ethical Black Box. This is a comprehensive logging mechanism that captures AI system inputs, outputs, intermediate signals, model states, and the reasoning (where available) behind decisions [71, 126, 154, 176]. Inspired by aviation black boxes, this mechanism provides a continuous, tamper-evident record of the AI system's behavior. Such detailed traceability is essential for post-incident analysis, forensic auditing, compliance verification, and diagnosing failures—particularly in high-stakes domains involving safety, fairness, or accountability [35]. By enabling stakeholders to reconstruct decision pathways, the Ethical Black Box strengthens transparency and supports ethical AI governance. It also facilitates proactive oversight by making deviations from expected behavior detectable long before they propagate into harmful system outcomes.

- Example: In an AI-driven financial system, an ethical black box records every loan-approval decision, including input features (e.g., credit score, income), model-inferred risk factors, confidence scores, and decision rationale. During audits or dispute resolution, this record allows regulators or internal teams to evaluate whether decisions were fair, explainable, and aligned with organizational and legal requirements.
- Related AITDs: Mitigates *Ethical Debt* by providing the transparency and traceability needed to assess fairness, detect discrimination, and verify decision legitimacy. Addresses *Entanglement* by offering clear logs that disentangle complex interactions between system components. Mitigates *Undeclared Consumers* by recording which systems or entities accessed or consumed AI outputs, ensuring accountability and preventing unauthorized use.

6.2.24 Human–AI Control Mode Switching. This guideline introduces a configurable mechanism that allows users or system operators to dynamically determine whether AI-generated outputs should be executed autonomously or treated as advisory suggestions requiring human confirmation before action [73, 123–125]. By enabling seamless switching between automated and human-in-the-loop modes, this guideline enhances operational transparency, prevents overreliance on automated decisions, and ensures that critical tasks remain under appropriate levels of human oversight. Such control mechanisms are especially vital in high-stakes or safety-critical environments—such as transportation, healthcare, finance, and industrial automation—where fully autonomous operation may expose the system to unacceptable levels of risk or error propagation. Human–AI control mode switching also provides an additional safeguard against unexpected system behaviors, model drift, or misinterpretations by ensuring that human judgment can override or validate AI outputs before execution.

- Example: In an autonomous vehicle, the mode-switching mechanism allows the driver to specify whether the AI should automatically initiate lane changes or merely suggest them, enabling manual review in dense traffic or

hazardous conditions. This ensures that the human operator retains situational awareness and can intervene when contextual nuances exceed the model's competence.

- Related AITDs: Mitigates *Entanglement* by providing a clear separation between AI-generated recommendations and their execution, reducing the risk of tightly coupled autonomous decisions leading to cascading failures. Also addresses *Undeclared Consumers* by ensuring that AI outputs are not silently consumed or acted upon without explicit human approval, thereby improving transparency and accountability in how AI-driven decisions are used.

6.2.25 Unified Programming Language Policy. Adopting a single, standardized programming language across the AI development lifecycle reduces fragmentation in implementation practices and streamlines system-wide security, testing, and audit processes [30, 131, 223]. When AI pipelines depend on multiple languages—such as Python for modeling, R for analytics, and Java or C++ for deployment—teams face increased complexity in dependency management, debugging, and vulnerability scanning. A unified programming language policy improves maintainability, enhances developer productivity, supports more consistent security hardening, and reduces integration overhead by minimizing cross-language interoperability challenges.

- Example: In an AI-based e-commerce recommendation system, standardizing all components—feature engineering, model training, batch scoring, and API serving—on Python enables consistent dependency management (e.g., pip/conda), uniform static analysis, and streamlined deployment pipelines, lowering the risk of language-specific vulnerabilities or mismatched runtime environments.
- Related AITDs: *Multiple Language Smells (MLS)* — reduces fragmentation and eliminates hard-to-maintain cross-language boundaries; *Glue Code Debt* — minimizes the need for connectors or wrappers between mismatched languages; *Compatibility Debt* — avoids inconsistencies in runtime environments and library ecosystems caused by heterogeneous language stacks.

6.2.26 AI-Adaptive Secure Design Practices. This guideline emphasizes the integration of AI-specific ethical, security, and operational requirements into established system design methodologies, such as Unified Modeling Language (UML), SysML, workflow diagrams, or architectural blueprints [61, 153, 180, 197]. Traditional design methods often fall short in representing the unique risks of AI components—such as model drift, data dependency volatility, adversarial susceptibility, or fairness constraints. By adapting these methods to explicitly model AI behaviors, data flows, decision boundaries, and governance constraints, organizations can surface potential vulnerabilities early in the design phase. This ensures that AI-driven components are not treated as black-box add-ons but as first-class citizens whose ethical, safety, and security requirements shape the architecture from its inception.

Embedding AI-aware design practices also supports cross-team alignment by enabling data engineers, ML researchers, and software architects to reason about system interactions holistically. Early modeling of AI-specific design concerns reduces downstream refactoring, minimizes architectural inconsistencies, and strengthens long-term system maintainability.

- Example: When architecting an AI-powered traffic management system, UML component diagrams may explicitly represent fairness constraints, model-update workflows, input-validation checkpoints, and monitoring hooks. Including these aspects early ensures the final system adheres to ethical expectations, safety constraints, and regulatory requirements.

- Related AITDs: *Design Debt* — mitigated by embedding robust architectural principles and ethical/security considerations from the outset; *Boundary Erosion* — reduced by clearly representing AI component boundaries and interfaces; *Configuration Debt* — minimized by documenting AI-specific configuration parameters within formal design artifacts; *Entanglement* — avoided by designing modular, well-isolated AI components with explicitly defined interactions.

RQ3.2: Security Guidelines for Mitigating AITDs

Through a systematic consolidation of industry best practices, standards, and research insights, the study proposes a comprehensive set of **26 security-oriented guidelines** designed to mitigate the risks associated with AI Technical Debt (AITD). These guidelines address vulnerabilities across the entire AI development and operational lifecycle, with emphasis on data pipelines, model training and deployment, system integration, and governance structures. The key thematic categories include:

- **Transparency and Accountability Mechanisms**: Ethical Black Boxes, auditable decision logs, and traceability tools to support post-hoc analysis and regulatory compliance.
- **Human–AI Oversight and Control**: Human–AI mode switching, human-on-the-loop and human-in-the-loop procedures for high-stakes decisions, and manual override pathways.
- **Continuous Monitoring and Drift Management**: Behavioral monitoring, data drift and concept drift detection, performance tracking, and anomaly detection pipelines.
- **Resilience and Continuity Engineering**: Multi-model consensus, redundancy and failover mechanisms, fallback policies, and robustness techniques against adversarial or unexpected system behavior.
- **Secure Data Handling and Governance**: Encryption, access control, data masking, secure storage, automated deletion routines, and standardized data formats to reduce data-related AITDs.
- **Model Security and Integrity**: Watermarking, hardened models, adversarial and defensive training, and integrity validation checks across the model lifecycle.
- **Threat Intelligence and Incident Response**: AI-specific threat modeling, red-teaming, alignment testing, and rapid incident response workflows tailored for AI-enabled systems.
- **Configuration, Dependency, and Resource Management**: Version control for AI components, environment reproducibility, AI SBOM management, modularization, and resource continuity planning to reduce operational AITD accumulation.

Overall Emphasis: The presented set of 26 guidelines aims to enhance system resilience, reduce vulnerability exposure, and ensure that AI-enabled systems remain secure, transparent, and trustworthy throughout their lifecycle. These guidelines form the foundation for mitigating the diverse forms of AITD identified in this study.

Table 8. Mapping of Security Guidelines to Related AI Technical Debts (AITDs)

S/N	Security Guideline	Related AITDs	Explanation
1	Configuration Management	Configuration Debt, Versioning Debt, Pipeline Jungle, Compatibility Debt	Ensures consistent, auditable, and controlled configurations across models, pipelines, dependencies, and environments.
2	Encryption and Secure Data Storage	Data Debt, Configuration Debt, Documentation Debt, Versioning Debt	Protects sensitive data via encryption at rest/in transit, secure storage, and proper cryptographic configuration.
3	Continuous Behavioral & Performance Monitoring	Ethical Debt, Configuration Debt, Data Debt, Hidden Feedback Loops, Design Debt	Detects drift, misconfigurations, unfair outcomes, and anomalous behaviors.
4	Incident Response for AI Systems	Hidden Feedback Loops, Unstable Data Dependencies, Configuration Debt, SATD	Enables structured detection, containment, and recovery from AI-specific incidents (e.g., poisoning).
5	Data Validation	Data Debt, Pipeline Jungle, Correction Cascades	Ensures input data is correct, consistent, authentic, and complete before entering AI pipelines.
6	Input Sanitization (Defense Against Evasion Attacks)	Algorithmic Inclination, Entanglement, Correction Cascades, Defect Debt	Removes adversarial inputs and malicious payloads to prevent evasion attacks.
7	Model Watermarking (Model Integrity Controls)	Undeclared Consumers, Versioning Debt, Documentation Debt, Glue Code	Detects tampering, unauthorized use, and illicit replication of models.
8	Use of Hardened Models & Defensive Training	Algorithmic Inclination, Entanglement, Correction Cascades, Design Debt	Improves robustness via adversarial training and defense-oriented modeling.
9	Red Teaming and AI Alignment (RLHF)	Ethical Debt, Algorithmic Inclination, Overly Simplified Metrics, SATD	Identifies vulnerabilities and aligns model behavior with safe and ethical outcomes.
10	Adequate AI Resource & Continuity Management	Configuration Debt, Process/Infrastructure Debt	Ensures AI systems have reliable resources and continuity plans for safe operation.
11	Secure Coding Practices	Glue Code, Duplicate Model Code, Unwanted Debugging Code, Defect Debt	Enforces secure development, dependency scanning, and removal of insecure code artifacts.
12	Data Leak Prevention (DLP)	Data Debt, Documentation Debt, Dispensable Dependency, Undeclared Consumers	Prevents accidental or malicious exfiltration of sensitive data.
13	Data Masking & Information Deletion	Data Debt, Unstable Data Dependencies, Process/Infrastructure Debt, Documentation Debt	Anonymizes and securely deletes sensitive/obsolete data to reduce risk.
14	Threat Intelligence	Dispensable Dependency, SML, Configuration Debt, Data Debt, Boundary Erosion, Compatibility Debt, Documentation/Versioning Debt	Provides early awareness of AI-specific threats and emerging vulnerabilities.
15	Human–AI Control Mode Switching	Entanglement, Undeclared Consumers	Enables human oversight, preventing unintended or unsafe autonomous decisions.
16	Multi-Model Consensus Decision Framework	Correction Cascades, Hidden Feedback Loops	Uses diverse models to detect inconsistencies and avoid cascading failures.
17	AI Redundancy & Failover Mechanisms	Algorithmic Inclination, Unstable Data Dependencies	Eliminates single-model or single-data-source failure points.
18	Ethical Black Box	Entanglement, Undeclared Consumers, Ethical Debt	Enhances traceability and accountability for AI decisions.
19	AI Component Divergence Monitoring	Correction Cascades, Unstable Data Dependencies	Detects divergence across redundant components.
20	Controlled AI Output Consumers	Undeclared Consumers, Glue Code, Documentation Debt	Prevents unauthorized or untracked consumption of AI outputs.
21	API-Governed AI Access Control	Undeclared Consumers, Dispensable Dependency, Glue Code, Versioning Debt	Enforces controlled access to AI components and outputs.
22	AI System Software Bill of Materials (AI-SBOM)	Data Debt, Dispensable Dependency, Versioning Debt, Documentation Debt	Tracks datasets, libraries, and model lineage for transparency and safety.
23	Modular & Reusable AI Components	Glue Code, Deep God File, Design Debt, Duplicate Model Code	Promotes modular designs that reduce coupling and simplify updates.
24	Standardized & Interoperable Data Formats	Glue Code, Unstable Data Dependencies, Compatibility Debt	Ensures interoperability and reduces risky conversions.
25	Unified Programming Language Policy	Multiple Language Smells (MLS), Glue Code, Compatibility Debt	Reduces complexity and lowers cross-language security risks.
26	AI-Adaptive Secure Design Practices	Design Debt, Boundary Erosion, Configuration Debt, Entanglement	Embeds ethical and security considerations directly into AI system design.

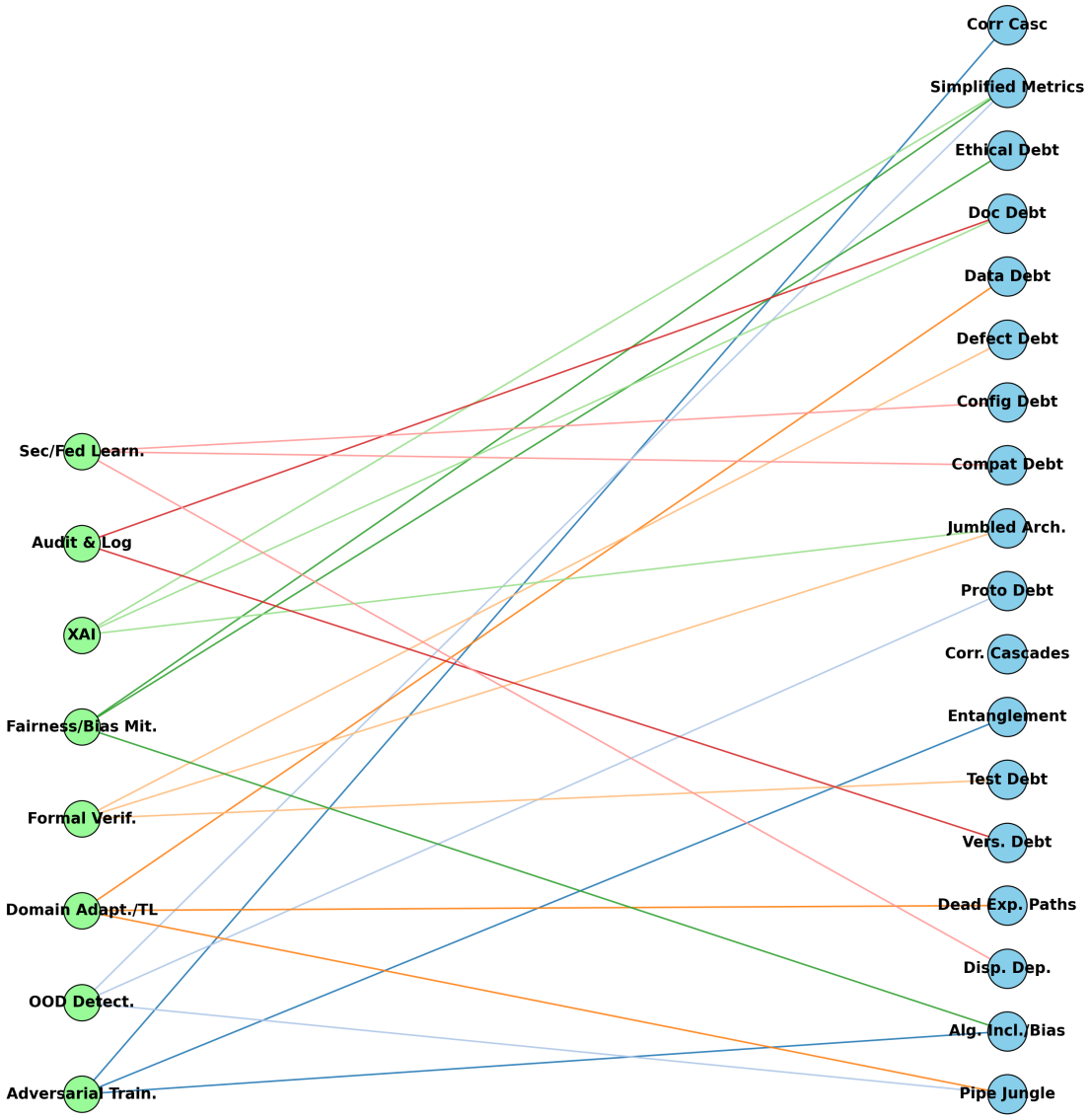

Fig. 6. **Mapping of Safety Guidelines to Related AI Technical Debts (AITDs).** This figure visualizes the relationships between key AI safety guidelines and the specific technical debts they help mitigate. Each guideline (green nodes) is connected to one or more AITDs (blue nodes) that pose risks to safety attributes such as explainability, robustness, fairness, and security.

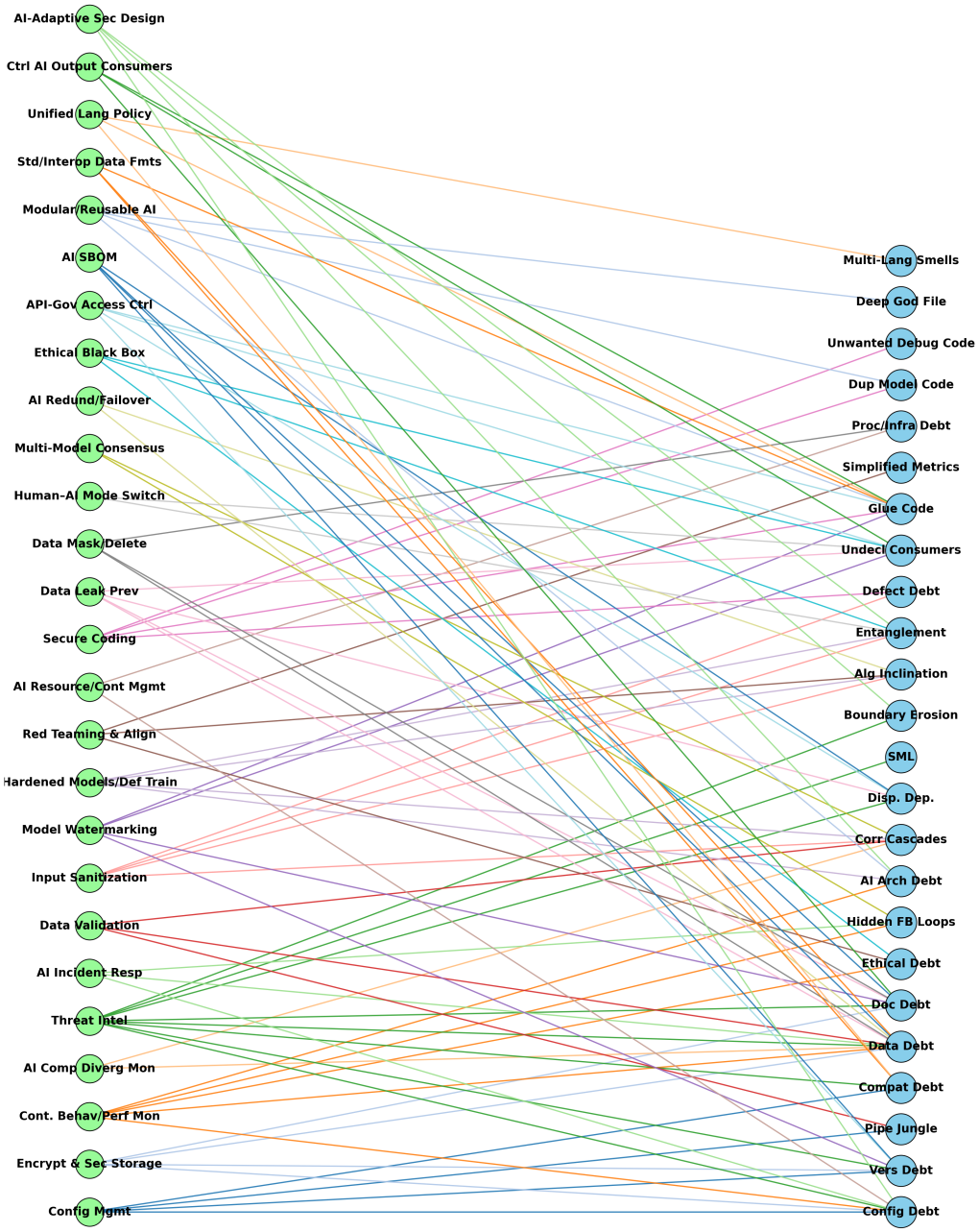

Fig. 7. **Mapping of Security Guidelines to Related AI Technical Debts (AITDs)**. This figure visualizes the relationships between key AI security guidelines and the specific technical debts they help mitigate. Each guideline (green node) is connected to one or more AITDs (blue nodes) that introduce vulnerabilities or risks to system security and reliability. The connections illustrate how implementing these guidelines can address multiple forms of AI Technical Debt—such as data, model, and configuration-related debts—thereby enhancing the overall resilience and trustworthiness of AI-enabled systems.

7 Principal findings

This systematic review consolidates evidence from 60 primary studies and identifies 31 distinct forms of technical debt as they manifest in AI-enabled systems. Using grounded theory and a root-cause–oriented synthesis approach, these debts were organized into a unified taxonomy consisting of seven major classes: (1) Data & Library–Related Debts, (2) Model & Code–Related Debts, (3) Algorithm–Related Debts, (4) Design & Architecture Debts, (5) Operational & Lifecycle Debts, (6) Documentation & Communication Debts, and (7) Testing & Quality Assurance Debts. This classification shifts the perspective from traditional categorizations toward a cause-driven view, enabling deeper analysis of how AITDs originate, propagate, and affect downstream system behavior, quality, and risk.

Root-Cause–Oriented Debt Landscape: The taxonomy reveals that AITDs arise from heterogeneous origins across the AI development pipeline. *Data & Library–Related Debts*, such as Data Debt and Pipeline Jungle, remain dominant and often triggered by incomplete, biased, outdated, or inconsistently processed datasets. These debts impede reproducibility and increase the likelihood of bias amplification and model drift. *Model & Code–Related Debts* capture issues such as Model Entanglement, Model Complexity, and Glue Code, which introduce maintainability bottlenecks and obscure system behavior. *Algorithm–Related Debts*, including Algorithmic Inclination or misaligned objective functions, reflect deeper issues with optimization choices or inadequately specified learning criteria. *Design & Architecture Debts* highlight structural shortcomings such as poor modularity, tightly coupled components, and undocumented dependencies, which impair scalability and evolution capability. *Operational & Lifecycle Debts* underscore challenges in deployment, monitoring, retraining, and model governance—areas increasingly recognized as critical in MLOps environments. Finally, *Documentation & Communication Debts* and *Testing & QA Debts* reveal persistent gaps in transparency, traceability, and verification, further weakening system accountability and reliability. Collectively, this taxonomy illustrates that AITDs originate from systemic and interconnected root causes spanning data, model internals, architectural decisions, operational workflows, and organizational practices.

Impact on System Quality Attributes: Across the reviewed studies, AITDs were found to substantially degrade qualities such as maintainability, performance, scalability, robustness, and efficiency. Debts related to data governance and model behavior often introduce latent risks—manifesting only during redeployment, domain shifts, or integration with new pipelines. Model Entanglement, for instance, impedes debugging and retraining, while missing or poor documentation complicates audits and compliance checks. The analysis also highlights strong interactions between AITDs and explainability, interpretability, and fairness—quality dimensions that carry unique significance in AI engineering. These impacts extend beyond typical software failures, affecting trustworthiness and societal acceptance of AI systems. Notably, the findings show that many AITDs have a cascading effect: errors in data processing propagate into models, which then propagate into downstream decisions, magnifying technical and ethical risks.

Safety and Security Integration: A core contribution of this study is the explicit mapping of the 31 AITDs to **6 safety** and **12 security** risks identified in the literature. Certain debts, such as Data Debt, Undeclared Consumers, and Correction Cascades, were repeatedly linked to vulnerabilities including adversarial susceptibility, data leakage, unreliable model reuse, and unstable behavior under distribution shifts. Ethical and documentation-related debts were associated with failures in accountability, auditability, and safe human oversight. Operational debts further compromise resilience by impairing monitoring, retraining, and incident response workflows. This mapping confirms that AITDs function not merely as engineering inefficiencies but as systemic vulnerabilities with direct implications for safety, compliance, and risk governance—highlighting the need to integrate AITD assessment with security engineering and safety certification processes.

Mitigation Strategies: The study synthesizes a total of **34 mitigation guidelines** - **8 safety guidelines** and **26 security** - designed to address the root causes and propagation pathways of AITDs. These include mechanisms such as: *Human–AI Control Mode Switching*, *Ethical Black Boxes for decision logging*, *Continuous Behavioral and Drift Monitoring*, *Model Watermarking and Hardening*, *Redundancy and Failover Architectures*, *Out-of-Distribution Detection*, and *Formal Verification and Model Assurance* for high-risk deployments. Each guideline is mapped to one or more debt classes (Tables 7–8), creating a traceable link between AITD type, underlying cause, associated risks, and appropriate mitigation techniques. By grounding mitigation in root causes, the framework supports proactive debt prevention during design, development, deployment, and continuous operation.

AITD-MAP Framework: Building on the taxonomy and mappings, the study introduces the AITD-MAP (Mapping AI Technical Debt: Types, Impact, and Guidelines) framework. AITD-MAP integrates three dimensions: (i) the root-cause-oriented AITD taxonomy, (ii) the identified impacts on software quality, safety, and security, and (iii) the 34 mitigation guidelines. This unified model enables practitioners to diagnose debts based on symptoms, trace their propagation across pipelines, evaluate associated risks, and select targeted mitigation strategies. It also offers a conceptual foundation for future automated tools for AITD detection, monitoring, and refactoring in MLOps ecosystems.

Prevalence and Prioritization: The prevalence analysis shows that *Data Debt/Unstable Data Dependencies* is the most frequently reported AITD, appearing in 27 of 60 studies (45%). This is followed by *Glue Code* (30%), *Test Debt* (28.33%), and *Documentation Debt* (25.33%). Other commonly reported debts include *Requirement Debt* (23.33%), *Design Debt* (21.67%), and *Configuration Debt* (20%). These results indicate that data-related and model/code-related debts dominate current AI development challenges, particularly those associated with unstable data pipelines, ad-hoc integrations, prototype code, and insufficient testing practices. The relatively high frequency of documentation, requirement, and design debts further reflects the ongoing struggle to maintain traceability, architectural clarity, and consistent communication in rapidly evolving AI workflows. Overall, the distribution of AITDs underscores that the most pervasive issues stem from the early stages of the AI pipeline (data and preprocessing), the implementation layer (model/code quality), and system-level coordination (requirements, documentation, and configuration). This prioritization highlights the need for context-aware debt management practices tailored to system maturity, domain criticality, and operational risk profiles.

Synthesis and Implications: Overall, this review provides the first root-cause–oriented, security-aware, and mitigation-driven characterization of AI Technical Debt. By connecting technical debt origin, systemic impact, and prescriptive mitigation, the study bridges a significant gap in the AI engineering literature. The findings emphasize that AITD management must be continuous, interdisciplinary, and transparent-spanning data governance, model design, architecture, testing, deployment, and organizational practices. Integrating AITD assessment into DevOps and MLOps workflows is essential for early detection and automated mitigation. This work lays a robust foundation for future empirical validations, tool development, and policy frameworks aimed at promoting sustainable, trustworthy, and resilient AI systems.

7.1 Strengths and limitations

This study demonstrates several notable strengths. It systematically identifies and categorizes 31 distinct types of AI Technical Debts (AITDs), offering a comprehensive taxonomy that captures their structural, operational, and ethical implications in AI-based systems. By emphasizing the intersection between AITDs and critical concerns—particularly security and safety—the study provides a nuanced understanding of how these debts affect system attributes such as maintainability, robustness, and resilience. The introduction of AITD-MAP (Mapping AI Technical Debt: Types, Impact,

and Guidelines) further strengthens the contribution by organizing insights across taxonomy, quality attributes, and mitigation strategies into a coherent, actionable framework.

Another key strength lies in the proposed mitigation strategies. The study presents 34 security and safety guidelines that are explicitly mapped to the identified AITDs, offering practical pathways to address risks in AI system development and deployment. This integration of mitigation guidance into a structured framework represents a novel step toward responsible and sustainable AI engineering. Additionally, the application of grounded theory ensures a rigorous and data-driven process for categorization, enhancing the validity and reliability of findings. The review draws on 60 primary studies from diverse domains, reinforcing the generalizability of the findings across various AI use cases and development contexts.

However, the study is not without limitations. The reliance on selected academic databases (ACM Digital Library, IEEE Xplore, Scopus, and Springer) may have led to the exclusion of relevant studies from other specialized or emerging sources. The manual nature of the selection, coding, and classification process—though mitigated by cross-checking and author consensus—still poses a risk of human bias or oversight. Furthermore, the study's focus on recent literature (2015–2025) offers a timely synthesis but may omit insights from earlier foundational works on technical debt or AI safety.

While the inclusion of mitigation guidelines is a major strength, their practical applicability remains to be validated through industrial case studies or empirical trials. Additionally, some emerging debts—such as Correction Cascades and Boundary Erosion—appear less frequently in the literature, making their classification provisional and highlighting the need for further empirical investigation. Despite these limitations, this research significantly advances the field by providing a taxonomy-driven, security-aware, and mitigation-oriented perspective on AI Technical Debt. It sets a solid foundation for future empirical studies and practical tool development aimed at improving the quality, accountability, and resilience of AI-based systems.

8 Conclusion and future work

As AI-based systems increasingly operate in safety-critical and trust-sensitive environments, the accumulation of AI Technical Debt (AITD) poses a serious challenge to their long-term sustainability, maintainability, and trustworthiness. This study conducted a systematic review of 60 primary studies and identified 31 distinct types of AITDs, offering a detailed taxonomy and analysis of their origin, impact, and mitigation strategies. A key contribution of this work is the adoption of AI TRiSM (AI Trust, Risk, and Security Management) as a conceptual lens to reinterpret AITDs through the dimensions of trust, risk, safety, and security. This perspective consolidates fragmented concerns across technical and governance domains and emphasizes the interdependence between safety and security debt—both of which are often treated in isolation in existing literature.

To operationalize this perspective, we proposed the AITD-MAP framework, which unifies taxonomic classification, impact assessment, and mitigation planning for AITDs. We further introduced 34 actionable guidelines to assist practitioners and researchers in identifying and addressing AITDs—especially those that compromise safety, security, and overall system trustworthiness. In summary, this study offers a structured, multi-dimensional understanding of AITDs and introduces a practical roadmap for addressing them. The proposed taxonomy, impact mappings, and mitigation framework contribute to the growing body of knowledge on responsible and sustainable AI engineering—supporting researchers, developers, and policymakers in building safe, secure, reliable, and maintainable AI systems.

While this study consolidates and maps a comprehensive set of safety and security guidelines to the identified AITDs, future work should focus on empirically validating these guidelines in industrial settings and evaluating their

effectiveness within MLOps pipelines. A natural extension of this work involves operationalizing the AITD-MAP framework into automated tools for debt detection, monitoring, and risk-aware refactoring. Such developments can draw on principles of responsible, safe, and secure AI engineering, as discussed in Section 6, with the aim of improving guideline applicability and supporting continuous assurance.

Moreover, although the reviewed debts primarily reflect challenges in foundational AI systems, the insights extend to emerging paradigms such as Agentic-AI [43, 181] and large language models (LLMs) [41]. These systems introduce increased autonomy, complexity, and dependency on dynamic data sources, which may intensify the accumulation and propagation of AITDs. Investigating how the proposed taxonomy, guidelines, and AITD-MAP framework apply to these new architectures represents a promising direction for future research.

8.1 Future Challenges and Research Directions

Building upon the findings and risk-informed roadmap proposed in this study, several open challenges and research directions emerge that warrant further exploration to advance the management of AI Technical Debt (AITD) in intelligent systems.

- **Standardized Metrics and Measurement Frameworks.** Although this study provides a taxonomy and qualitative mapping of AITDs, the absence of standardized metrics for assessing the severity, propagation, and resolution of such debts remains a critical gap. Future research should focus on defining quantifiable indicators and benchmark datasets to enable empirical measurement of AITD impact on software quality, maintainability, and trustworthiness.
- **Tooling and Automation for Debt Detection.** While mature automated tools already exist for detecting several forms of technical debt—particularly code- and design-related debts - many AI-specific and cross-cutting AITDs (e.g., data-related debt, pipeline-level debt, and socio-technical debt) still rely heavily on manual qualitative analysis. Developing AI-aware and integrated detection and monitoring approaches-leveraging static analysis, ML-driven pattern recognition, and natural language processing of development artifacts-will therefore be crucial for operationalizing comprehensive AITD management within MLOps and continuous integration pipelines.
- **Dynamic Risk Modeling and Simulation.** As AITDs evolve over time through feedback loops and system updates, static analyses may fail to capture their long-term implications. Future studies should adopt dynamic simulation techniques (e.g., digital twins, system dynamics) to model how debts accumulate, interact, and impact reliability, security, and ethics throughout the system lifecycle.
- **Integration with AI Governance and TRiSM Frameworks.** The mapping with AI TRiSM and related governance models revealed strong conceptual overlap between AITDs and concerns such as fairness, transparency, and accountability. Future work should formalize these relationships by embedding debt-aware risk assessment modules within governance, assurance, and certification frameworks.
- **Towards a Debt-Aware Software Engineering Paradigm.** The long-term vision is to establish a debt-aware AI engineering paradigm where design decisions, testing strategies, and deployment pipelines are continuously informed by AITD risk indicators. Future work should develop integrated frameworks that connect technical debt management with responsible and sustainable AI system development.

By addressing these open challenges, future research can move toward a quantitative, automated, and context-aware understanding of AI Technical Debt. This will strengthen both theoretical foundations and practical interventions, ensuring the responsible and sustainable evolution of intelligent systems.

References

[1] Nikolaj Aagaard, Eske K Aasvang, and Christian S Meyhoff. 2024. Discrepancies between Promised and Actual AI Capabilities in the Continuous Vital Sign Monitoring of In-Hospital Patients: A Review of the Current Evidence. *Sensors (Basel, Switzerland)* 24, 19 (2024), 6497.

[2] Amina Adadi and Mohammed Berrada. 2018. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access* 6 (2018), 52138–52160. doi:10.1109/ACCESS.2018.2870052

[3] Nilay Yorgancılar Akgül, Tuğba Taşkaya Temizel, Özden Özcan Top, and Pelin Dayan Akman. 2025. Aligning Data Debt with AI-Integrated Software Project Lifecycle Processes: A Standard-Based Mapping Approach. In *2025 IEEE/ACM International Conference on Technical Debt (TechDebt)*. IEEE, 1–11.

[4] Pelin Dayan Akman, Özden Özcan Top, and Tugba Taskaya Temizel. 2025. People and Management Debt in ML-Integrated Software Projects: Structuring Industry Insights. *IEEE Access* (2025).

[5] Mohammed Naveed Akram, Akshatha Ambekar, Ioannis Sorokos, Koorosh Aslansefat, and Daniel Schneider. 2022. StaDRe and StaDRo: reliability and robustness estimation of ML-based forecasting using statistical distance measures. In *International Conference on Computer Safety, Reliability, and Security*. Springer, 289–301.

[6] Md Abdullah Al Alamin and Gias Uddin. 2021. Quality assurance challenges for machine learning software applications during software development life cycle phases. In *2021 IEEE International Conference on Autonomous Systems (ICAS)*. IEEE, 1–5.

[7] Mohannad Alahdab and Gül Çalıklı. 2019. 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, 195–202.

[8] A.S. Albahri, Ali M. Duhaim, Mohammed A. Fadhel, Alhamzah Alnoor, Noor S. Baqer, Laith Alzubaidi, O.S. Albahri, A.H. Alamoodi, Jinshuai Bai, Asma Salhi, Jose Santamaría, Chun Ouyang, Ashish Gupta, Yuantong Gu, and Muhammet Deveci. 2023. A systematic review of trustworthy and explainable artificial intelligence in healthcare: Assessment of quality, bias risk, and data fusion. *Information Fusion* 96 (2023), 156 – 191. Cited by: 359. doi:10.1016/j.inffus.2023.03.008

[9] Danyllo Albuquerque, Everton Guimaraes, Graziela Tonin, Mirko Perkusich, Hyggo Almeida, and Angelo Perkusich. 2022. Comprehending the use of intelligent techniques to support technical debt management. In *Proceedings of the International Conference on Technical Debt*. 21–30.

[10] Nastoska Aleksandra, Jancheska Bojana, Rizinski Maryan, and Trajanov Dimitar. 2025. Evaluating Trustworthiness in AI: Risks, Metrics, and Applications Across Industries. *Electronics* 14, 13 (2025), 2717.

[11] Reem Alfayez, Wesam Alwehaibi, Robert Winn, Elaine Venson, and Barry Boehm. 2020. A systematic literature review of technical debt prioritization. In *Proceedings of the 3rd international conference on technical debt*. 1–10.

[12] Muhammad Ali, Yim-Fun Hu, Doanh Kim Luong, George Oguntala, Jian-Ping Li, and Kanaan Abdo. 2020. Adversarial Attacks on AI based Intrusion Detection System for Heterogeneous Wireless Communications Networks. In *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*. 1–6. doi:10.1109/DASC50938.2020.9256597

[13] Sultan Alneyadi, Elankayer Sithirasenan, and Vallipuram Muthukkumarasamy. 2016. A survey on data leakage prevention systems. *Journal of Network and Computer Applications* 62 (2016), 137–152.

[14] Nicolli SR Alves, Thiago S Mendes, Manoel G De Mendonça, Rodrigo O Spínola, Forrest Shull, and Carolyn Seaman. 2016. Identification and management of technical debt: A systematic mapping study. *Information and Software Technology* 70 (2016), 100–121.

[15] Areti Ampatzoglou, Apostolos Ampatzoglou, Alexander Chatzigeorgiou, and Paris Avgeriou. 2015. The financial aspect of managing technical debt: A systematic literature review. *Information and Software Technology* 64 (2015), 52–73.

[16] Giusy Annunziata, Stefano Lambiase, Damian A Tamburri, Willem-Jan Van Den Heuvel, Fabio Palomba, Gemma Catolino, Filomena Ferrucci, and Andrea De Lucia. 2025. Uncovering community smells in machine learning-enabled systems: Causes, effects, and mitigation strategies. *ACM Transactions on Software Engineering and Methodology* 34, 6 (2025), 1–48.

[17] Fabio Arnez, Huascar Espinoza, Ansgar Radermacher, and François Terrier. 2021. Improving Robustness of Deep Neural Networks for Aerial Navigation by Incorporating Input Uncertainty. In *Computer Safety, Reliability, and Security. SAFECOMP 2021 Workshops*, Ibrahim Habli, Mark Sujan, Simos Gerasimou, Erwin Schoitsch, and Friedemann Bitsch (Eds.). Springer International Publishing, Cham, 219–225.

[18] Fabio Arnez, Huascar Espinoza, Ansgar Radermacher, and François Terrier. 2021. Improving robustness of deep neural networks for aerial navigation by incorporating input uncertainty. In *International Conference on Computer Safety, Reliability, and Security*. Springer, 219–225.

[19] Anders Arpteg, Bjørn Brinne, Luka Crnkovic-Friis, and Jan Bosch. 2018. Software engineering challenges of deep learning. In *2018 44th euromicro conference on software engineering and advanced applications (SEAA)*. IEEE, 50–59.

[20] Paris Avgeriou, Philippe Kruchten, Ipek Ozkaya, and Carolyn Seaman. 2016. Managing technical debt in software engineering (dagstuhl seminar 16162). *Dagstuhl reports* 6, 4 (2016), 110–138.

[21] Litan Avivah. 2024. Tackling Trust, Risk and Security in AI Models. *https://www.gartner.com/en/articles/ai-trust-and-ai-risk* (2024).

[22] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862* (2022).

[23] Sourav Banerjee, Mathews Thomas, Vinod Chavan, Utpal Mangla, and Srinivas Tummalapenta. 2025. Securing the future of AI: a holistic approach to trust and robustness. In *Assurance and Security for AI-enabled Systems 2025*, Vol. 13476. SPIE, 121–142.

[24] Efe Barlas, Xin Du, and James C Davis. 2022. Exploiting input sanitization for regex denial of service. In *Proceedings of the 44th International Conference on Software Engineering*. 883–895.

[25] Gabriele Bavota and Barbara Russo. 2016. A large-scale empirical study on self-admitted technical debt. In *Proceedings of the 13th international conference on mining software repositories*. 315–326.

[26] Hrvoje Belani, Marin Vukovic, and Željka Car. 2019. Requirements engineering challenges in building AI-based complex systems. In *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*. IEEE, 252–255.

[27] Gueltoum Bendiab, Amina Hameurlaine, Georgios Germanos, Nicholas Kolokotronis, and Stavros Shiaeles. 2023. Autonomous vehicles security: Challenges and solutions using blockchain and artificial intelligence. *IEEE Transactions on Intelligent Transportation Systems* 24, 4 (2023), 3614–3637.

[28] Aaditya Bhatia, Foutse Khomh, Bram Adams, and Ahmed E Hassan. 2023. An Empirical Study of Self-Admitted Technical Debt in Machine Learning Software. *arXiv preprint arXiv:2311.12019* (2023).

[29] Goutham Bilakanti. [n. d.]. Secure Data Masking for Healthcare Data Protection. ([n. d.]).

[30] Justus Bogner, Roberto Verdecchia, and Ilias Gerostathopoulos. 2021. Characterizing technical debt and antipatterns in AI-based systems: A systematic mapping study. In *2021 IEEE/ACM International Conference on Technical Debt (TechDebt)*. IEEE, 64–73.

[31] Gusseppe Bravo-Rocca, Peini Liu, Jordi Guitart, Ajay Dholakia, David Ellison, and Miroslav Hodak. 2022. Human-in-the-loop online multi-agent approach to increase trustworthiness in ML models through trust scores and data augmentation. In *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 32–37.

[32] Eric Breck, Shanqing Cai, Eric Nielsen, Michael Salib, and D Sculley. 2017. 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, 1123–1132.

[33] Alessio Bucaioni, Rick Kazman, and Patrizio Pelliccione. 2025. A checklist of quality concerns for architecting ML-intensive systems. *Journal of Systems and Software* (2025), 112612.

[34] Anna L Buczak, Benjamin D Baugher, Adam J Berlier, Kayla E Scharfstein, and Christine S Martin. 2022. Explainable forecasts of disruptive events using recurrent neural networks. In *2022 IEEE international conference on assured autonomy (ICAA)*. IEEE, 64–73.

[35] Jean-Christophe Bélisle-Pipon, Erica Monteferrante, Marie-Christine Roy, and Vincent Couture. 2023. Artificial intelligence ethics has a black box problem. *AI and Society* 38, 4 (2023), 1507 – 1522. Cited by: 32. doi:10.1007/s00146-021-01380-0

[36] Rojas Camilo, Sato Yuki, and Bennett Eleanor. 2024. AI-DRIVEN THREAT INTELLIGENCE: ENHANCING CYBERSECURITY IN MODERN SOFTWARE SYSTEMS. *Journal of Adaptive Learning Technologies* 1, 8 (2024), 53–68.

[37] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*. Ieee, 39–57.

[38] Albana Celepija, Bruno Lepri, and Raman Kazhamiakin. 2025. Towards a structured AI development lifecycle for reusable AI products in the public sector. (2025).

[39] Vinay Chamola, Vikas Hassija, A Razia Sulthana, Debshishu Ghosh, Divyansh Dhingra, and Biplab Sikdar. 2023. A review of trustworthy and explainable artificial intelligence (xai). *IEEe Access* 11 (2023), 78994–79015.

[40] Jiyoo Chang and Christine Custis. 2022. Understanding implementation challenges in machine learning documentation. In *Proceedings of the 2nd ACM Conference on Equity and Access in Algorithms, Mechanisms, and Optimization*. 1–8.

[41] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology* 15, 3 (2024), 1–45.

[42] Dev Kumar Chaudhary, Sandeep Srivastava, and Vikas Kumar. 2018. A review on hidden debts in machine learning systems. In *2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT)*. IEEE, 619–624.

[43] Chhavi Chawla, Siddharth Chatterjee, Sanketh Siddanna Gadadinni, Pulkit Verma, and Sourav Banerjee. 2024. Agentic AI: The building blocks of sophisticated AI business applications. *Journal of AI, Robotics & Workplace Automation* 3, 3 (2024), 1–15.

[44] Junjie Chen, Yihua Liang, Qingchao Shen, Jiajun Jiang, and Shuochuan Li. 2023. Toward understanding deep learning framework bugs. *ACM Transactions on Software Engineering and Methodology* 32, 6 (2023), 1–31.

[45] Krishna Keerthi Chennam, Swapna Mudrakola, V. Uma Maheswari, Rajanikanth Aluvalu, and K. Gangadhara Rao. 2023. *Black Box Models for eXplainable Artificial Intelligence*. Springer International Publishing, Cham, 1–24. doi:10.1007/978-3-031-12807-3_1

[46] Dinesh Reddy Chirra. 2020. AI-Based Real-Time Security Monitoring for Cloud-Native Applications in Hybrid Cloud Environments. *Revista de Inteligencia Artificial en Medicina* 11, 1 (2020), 382–402.

[47] Seok-Hwan Choi, Jin-Myeong Shin, Peng Liu, and Yoon-Ho Choi. 2022. ARGAN: Adversarially robust generative adversarial networks for deep neural networks against adversarial examples. *IEEE Access* 10 (2022), 33602–33615.

[48] Jessica Cooper, Ognjen Arandjelović, and David J Harrison. 2022. Believe the HiPe: Hierarchical perturbation for fast, robust, and model-agnostic saliency mapping. *Pattern Recognition* 129 (2022), 108743.

[49] Juliet M Corbin and Anselm Strauss. 1990. Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative sociology* 13, 1 (1990), 3–21. doi:10.1007/BF00988593

[50] Davide Corsi, Enrico Marchesini, and Alessandro Farinelli. 2021. Formal verification of neural networks for safety-critical tasks in deep reinforcement learning. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (Proceedings of Machine Learning Research, Vol. 161)*, Cassio de Campos and Marloes H. Maathuis (Eds.). PMLR, 333–343. https://proceedings.mlr.press/v161/corsi21a.html

[51] Dolors Costal, Cristina Gómez, Santiago del Rey, and Silverio Martínez-Fernández. 2024. Using Metrics for Code Smells of ML Pipelines. In *2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C)*. 289–294. doi:10.1109/ICSA-C63560.2024.00055

[52] Pierre-Olivier Côté, Amin Nikanjam, Rached Bouchoucha, Ilan Basta, Mouna Abidi, and Foutse Khomh. 2024. Quality issues in machine learning software systems. *Empirical Software Engineering* 29, 6 (2024), 1–47.

[53] Warteruzannan Soyer Cunha, Guisella Angulo Armijo, and Valter Vieira de Camargo. 2020. Investigating non-usually employed features in the identification of architectural smells: A machine learning-based approach. In *Proceedings of the 14th Brazilian Symposium on Software Components, Architectures, and Reuse*. 21–30.

[54] Ward Cunningham. 1992. The WyCash portfolio management system. *ACM Sigplan Oops Messenger* 4, 2 (1992), 29–30.

[55] Guangye Dai, Saurav Sthapit, Gregory Epiphaniou, and Carsten Maple. 2021. Artificial intelligence technologies in building resilient machine learning. In *Competitive Advantage in the Digital Economy (CADE 2021)*, Vol. 2021. IET, 50–55.

[56] Ning Dang, Keyong Shao, Long Chen, and Min Yang. 2022. Multi-model decision-making seizure types classification based on transfer learning. *Proceedings - 2022 International Symposium on Control Engineering and Robotics, ISCER 2022* (2022), 192 – 201. Cited by: 6. doi:10.1109/ISCER55570.2022.00040

[57] Manas Dave and Neil Patel. 2023. Artificial intelligence in healthcare and education. *British dental journal* 234, 10 (2023), 761–764.

[58] Evren Dağlarli. 2020. Explainable Artificial Intelligence (xAI) Approaches and Deep Meta-Learning Models. In *Advances and Applications in Deep Learning*, Marco Antonio Aceves-Fernandez (Ed.). IntechOpen, Rijeka, Chapter 5. doi:10.5772/intechopen.92172

[59] Jip WTM de Kok, Miguel Á Armengol de la Hoz, Ymke de Jong, Véronique Brokke, Paul WG Elbers, Patrick Thoral, Alejandro Castillejo, Tomás Trenor, Jose M Castellano, Alberto E Bronchalo, et al. 2023. A guide to sharing open healthcare data under the General Data Protection Regulation. *Scientific data* 10, 1 (2023), 404.

[60] Ronnie de Souza Santos, Felipe Fronchetti, Sávio Freire, and Rodrigo Spinola. 2025. Software fairness debt: Building a research agenda for addressing bias in AI systems. *ACM Transactions on Software Engineering and Methodology* 34, 5 (2025), 1–21.

[61] Shouki A Ebad. 2022. Exploring how to apply secure software design principles. *IEEE Access* 10 (2022), 128983–128993.

[62] Neil Ernst, Rick Kazman, and Julien Delange. 2021. *Technical Debt in Practice: How to Find It and Fix It.* MIT Press.

[63] Marie Farrell, Matt Luckcuck, Laura Pullum, Michael Fisher, Ali Hessami, Danit Gal, Zvikomborero Murahwi, and Ken Wallace. 2021. Evolution of the IEEE P7009 standard: Towards fail-safe design of autonomous systems. In *2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 401–406.

[64] Michael Feffer, Anusha Sinha, Wesley H Deng, Zachary C Lipton, and Hoda Heidari. 2024. Red-teaming for generative AI: Silver bullet or security theater?. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, Vol. 7. 421–437.

[65] Jean Feng, Rachael V. Phillips, Ivana Malenica, Andrew Bishara, Alan E. Hubbard, Leo A. Celi, and Romain Pirracchio. 2022. Clinical artificial intelligence quality improvement: towards continual monitoring and updating of AI algorithms in healthcare. *npj Digital Medicine* 5, 1 (2022). Cited by: 175; All Open Access, Gold Open Access, Green Open Access. doi:10.1038/s41746-022-00611-y

[66] Emilio Ferrara. 2023. Fairness and bias in artificial intelligence: A brief survey of sources, impacts, and mitigation strategies. *Sci* 6, 1 (2023), 3.

[67] Michael Fisher, Viviana Mascardi, Kristin Yvonne Rozier, Bernd-Holger Schlingloff, Michael Winikoff, and Neil Yorke-Smith. 2021. Towards a framework for certification of reliable autonomous systems. *Autonomous Agents and Multi-Agent Systems* 35 (2021), 1–65.

[68] Harald Foidl and Michael Felderer. 2019. Risk-based data validation in machine learning-based software systems. In *proceedings of the 3rd ACM SIGSOFT international workshop on machine learning techniques for software quality evaluation*. 13–18.

[69] Harald Foidl, Michael Felderer, and Stefan Biffl. 2019. Technical debt in data-intensive software systems. In *2019 45th Euromicro conference on software engineering and advanced applications (SEAA)*. IEEE, 338–341.

[70] Harald Foidl, Michael Felderer, and Rudolf Ramler. 2022. 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*. 229–239.

[71] Valentina Franzoni. 2023. From black box to glass box: advancing transparency in artificial intelligence systems for ethical and trustworthy AI. In *International Conference on Computational Science and Its Applications*. Springer, 118–130.

[72] Narayana Gaddam. 2024. AI-POWERED DATA MASKING FOR PRIVACY-PRESERVING CLOUD DATA SHARING. *International Journal of Advanced Research in Cloud Computing* 5, 2 (2024), 12–22.

[73] Ming-Feng Ge, Jing-Zhe Xu, Zhi-Wei Liu, and Jian Huang. 2024. A mode-switched control architecture for human-in-the-loop teleoperation of multislave robots via data-training-based observer. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 54, 4 (2024), 2471–2483.

[74] Jiri Gesi, Siqi Liu, Jiawei Li, Iftekhar Ahmed, Nachiappan Nagappan, David Lo, Eduardo Santana de Almeida, Pavneet Singh Kochhar, and Lingfeng Bao. 2022. Code Smells in Machine Learning Systems. (2022).

[75] Julien Girard-Satabin, Michele Alberti, François Bobot, Zakaria Chihani, and Augustin Lemesle. 2022. CAISAR: A platform for Characterizing Artificial Intelligence Safety and Robustness. In *AISafety*.

[76] Mansi Girdhar, Junho Hong, and John Moore. 2023. Cybersecurity of autonomous vehicles: A systematic literature review of adversarial attacks and defense models. *IEEE Open Journal of Vehicular Technology* 4 (2023), 417–437.

[77] Paolo Giudici and Emanuela Raffinetti. 2023. SAFE Artificial Intelligence in finance. *Finance Research Letters* 56 (2023), 104088.

[78] Kseniia Gnitko. 2024. Systematic overview of AI security standards. *Available at SSRN 4922592* (2024).

[79] David Groombridge et al. 2022. Gartner top 10 strategic technology trends for 2023. *https://www. gartner. com/en/articles/gartner-top-10-strategic-technology-trends-for-2023* (2022).

[80] Praveen Gujar. 2025. Data standardization and interoperability. In *Data usability in the enterprise: how usability leads to optimal digital experiences*. Springer, 89–110.

[81] Jiajia Guo, Shaodan Ma, Chao-Kai Wen, and Shi Jin. 2025. Performance Monitoring-Enabled Reliable AI-Based CSI Feedback. *IEEE Transactions on Wireless Communications* 24, 1 (2025), 197 – 212. Cited by: 1. doi:10.1109/TWC.2024.3490600

[82] Pooja Gupta, Ravinder Singh, Harshdeep Kaur, N Subramanian, Neha Jain, et al. 2025. Optimizing Cryptocurrency Trading Strategies through Artificial Intelligence and Blockchain Integration: A Multi-Model Framework for Predictive Analytics. *Advances in Consumer Research* 2, 3 (2025).

[83] Sameeksha Gupta. 2025. GPU Reliability in AI Clusters: A Study of Failure Modes and Effects. *Journal Of Engineering And Computer Sciences* 4, 6 (2025), 298–306.

[84] Bálint Gyevnár and Atoosa Kasirzadeh. 2025. AI safety for everyone. *Nature Machine Intelligence* 7, 4 (2025), 531 – 542. Cited by: 0. doi:10.1038/s42256-025-01020-y

[85] Adib Habbal, Mohamed Khalif Ali, and Mustafa Ali Abuzaraida. 2024. Artificial Intelligence Trust, risk and security management (AI trism): Frameworks, applications, challenges and future research directions. *Expert Systems with Applications* 240 (2024), 122442.

[86] Thilo Hagendorff. 2021. Linking human and machine behavior: A new approach to evaluate training data quality for beneficial machine learning. *Minds and Machines* 31, 4 (2021), 563–593.

[87] Tom Haider, Karsten Roscher, Felippe Schmoeller da Roza, and Stephan Günnemann. 2023. Out-of-Distribution Detection for Reinforcement Learning Agents with Probabilistic Dynamics Models. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS* 2023-May (2023), 851 – 859. Cited by: 15. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85171291933&partnerID=40&md5=eca3fa8f2d003fd46567c10148750abc

[88] Ronan Hamon, Henrik Junklewitz, Josep Soler Garrido, and Ignacio Sanchez. 2024. Three challenges to secure AI systems in the context of AI regulations. *Ieee Access* 12 (2024), 61022–61035.

[89] Moritz Hardt, Eric Price, and Nathan Srebro. 2016. Equality of opportunity in supervised learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems* (Barcelona, Spain) *(NIPS'16)*. Curran Associates Inc., Red Hook, NY, USA, 3323–3331.

[90] Hongmei He, John Gray, Angelo Cangelosi, Qinggang Meng, T Martin McGinnity, and Jörn Mehnen. 2021. The challenges and opportunities of human-centered AI for trustworthy robots and autonomous systems. *IEEE Transactions on Cognitive and Developmental Systems* 14, 4 (2021), 1398–1412.

[91] Elena Hernández, Mehmet Öztürk, Inés Sittón, and Sara Rodríguez. 2019. Data protection on FinTech platforms. In *Highlights of Practical Applications of Survivable Agents and Multi-Agent Systems. The PAAMS Collection: International Workshops of PAAMS 2019, Ávila, Spain, June 26–28, 2019, Proceedings 17*. Springer, 223–233.

[92] Kalle Hjerppe, Jukka Ruohonen, and Ville Leppänen. 2019. The general data protection regulation: Requirements, architectures, and constraints. *Proceedings of the IEEE International Conference on Requirements Engineering* 2019-September (2019), 265 – 275. Cited by: 39; All Open Access, Green Open Access. doi:10.1109/RE.2019.00036

[93] Anh Hoang and Hien Phan. 2024. Explainable AI in Finance: an Overview. (2024).

[94] Ben Hutchinson, Andrew Smart, Alex Hanna, Emily Denton, Christina Greer, Oddur Kjartansson, Parker Barnes, and Margaret Mitchell. 2021. Towards accountability for machine learning datasets: Practices from software engineering and infrastructure. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. 560–575.

[95] IBM. 2025. What is AI TRiSM? Accessed: 16 January 2026. https://www.ibm.com/think/topics/ai-trism

[96] ISO/IEC 27002. 2022. *List of ISO 27002:2022 Controls — What Changed in 2022?* Accessed: Dec 02, 2025.

[97] M Jaeyalakshmi, P Rohit Gangadhar, M Srivatsan, and M Bhavani. 2023. A Self-learning Ai-Based Information Leak Protection System. In *International Conference on Advances in Artificial Intelligence and Machine Learning in Big Data Processinging*. Springer, 68–78.

[98] Nikita Jaipuria, Katherine Stevo, Xianling Zhang, Meghana L Gaopande, Ian Calle, Jinesh Jain, and Vidya N Murali. 2022. deepPIC: Deep perceptual image clustering for identifying bias in vision datasets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4793–4802.

[99] Shadi Jawhar, Jeremy Miller, and Zeina Bitar. 2024. AI-based cybersecurity policies and procedures. In *2024 IEEE 3rd International Conference on AI in Cybersecurity (ICAIC)*. IEEE, 1–5.

[100] Hadhemi Jebnoun, Md Saidur Rahman, Foutse Khomh, and Biruk Asmare Muse. 2022. Clones in deep learning code: what, where, and why? *Empirical Software Engineering* 27, 4 (2022), 84.

[101] Gavin Jones, Dimitrios Kasimatis, Nikolaos Pitropakis, Richard Macfarlane, and William J Buchanan. 2025. Analysing the role of LLMs in cybersecurity incident management. *International Journal of Information Security* 24, 6 (2025), 1–14.

[102] Uday Kamath, John Liu, and James Whitaker. 2019. *Transfer Learning: Domain Adaptation*. Springer International Publishing, Cham, 495–535. doi:10.1007/978-3-030-14596-5_11

[103] Ryo Kamoi and Kei Kobayashi. 2020. Out-of-Distribution Detection with Likelihoods Assigned by Deep Generative Models Using Multimodal Prior Distributions. In *SafeAI@AAAI*. https://api.semanticscholar.org/CorpusID:212419400

[104] Stefan Katzenbeisser, Ilia Polian, Francesco Regazzoni, and Marc Stöttinger. 2019. Security in autonomous systems. In *2019 IEEE European Test Symposium (ETS)*. IEEE, 1–8.

[105] Omkar Khanvilkar, Mohamed Wiem Mkaouer, Eman Abdullah AlOmar, Abdelrahman ElSaid, Amal Chaaben, and Mohamed Touati. 2025. Automated Identification of Machine Learning Technical Debt Code Comments. In *2025 International Conference on Emerging Technologies and Computing (IC_ETC)*. IEEE, 1–6.

[106] Anton Khritankov. 2021. Hidden feedback loops in machine learning systems: A simulation model and preliminary results. In *Software Quality: Future Perspectives on Software Engineering Quality: 13th International Conference, SWQD 2021, Vienna, Austria, January 19–21, 2021, Proceedings 13*. Springer, 54–65.

[107] Jin-Young Kim and Sung-Bae Cho. 2020. Fair representation for safe artificial intelligence via adversarial learning of unbiased information bottleneck.. In *SafeAI@ AAAI*. 105–112.

[108] MNV Kiranbabu, A Jeraldine Viji, Amit Kumar Chandanan, Vijay Birchha, Tushar Kumar Pandey, and Sumit Kumar Sar. 2025. The Challenge of Adversarial Attacks on AI-Driven Cybersecurity Systems. *Journal of Cybersecurity & Information Management* 15, 1 (2025).

[109] Howard Kleinwaks, Ann Batchelor, and Thomas H Bradley. 2023. An Ontology for Technical Debt in Systems Engineering. *IEEE Open Journal of Systems Engineering* (2023).

[110] Howard Kleinwaks, Ann Batchelor, and Thomas H Bradley. 2023. Technical debt in systems engineering—A systematic literature review. *Systems Engineering* 26, 5 (2023), 675–687.

[111] Agneza Krajna, Mihael Kovac, Mario Brcic, and Ana Šarčević. 2022. Explainable artificial intelligence: An updated perspective. In *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO)*. IEEE, 859–864.

[112] Nina Kshetry and Lav R Varshney. 2019. Safety in the face of unknown unknowns: Algorithm fusion in data-driven engineering systems. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 8162–8166.

[113] P Vishnu Kumar, Tanaya Ganguly, Rolly Gupta, Kiran Sree Pokkuluri, Avinash Kumar V Mishra, and V Selvi. 2024. ML and AI Based Healthcare Model to more Interpretable and Transparent in Medical Diagnosis. *African Journal of Biological Sciences* (2024).

[114] Valentina Lenarduzzi, Francesco Lomio, Sergio Moreschini, Davide Taibi, and Damian Andrew Tamburri. 2021. Software quality for ai: Where we are now?. In *Software Quality: Future Perspectives on Software Engineering Quality: 13th International Conference, SWQD 2021, Vienna, Austria, January 19–21, 2021, Proceedings 13*. Springer, 43–53.

[115] Bo Li, Peng Qi, Bo Liu, Shuai Di, Jingen Liu, Jiquan Pei, Jinfeng Yi, and Bowen Zhou. 2023. Trustworthy AI: From principles to practices. *Comput. Surveys* 55, 9 (2023), 1–46.

[116] Yikun Li, Mohamed Soliman, and Paris Avgeriou. 2022. Identifying self-admitted technical debt in issue tracking systems using machine learning. *Empirical Software Engineering* 27, 6 (2022), 131.

[117] Yikun Li, Mohamed Soliman, and Paris Avgeriou. 2023. Automatic identification of self-admitted technical debt from four different sources. *Empirical Software Engineering* 28, 3 (2023), 65.

[118] Yikun Li, Mohamed Soliman, Paris Avgeriou, and Maarten Van Ittersum. 2023. DebtViz: A Tool for Identifying, Measuring, Visualizing, and Monitoring Self-Admitted Technical Debt. In *2023 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 558–562.

[119] Zengyang Li, Paris Avgeriou, and Peng Liang. 2015. A systematic mapping study on technical debt and its management. *Journal of Systems and Software* 101 (2015), 193–220.

[120] Jiakun Liu, Qiao Huang, Xin Xia, Emad Shihab, David Lo, and Shanping Li. 2020. Is using deep learning frameworks free? characterizing technical debt in deep learning frameworks. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Society*. 1–10.

[121] Jiakun Liu, Qiao Huang, Xin Xia, Emad Shihab, David Lo, and Shanping Li. 2021. An exploratory study on the introduction and removal of different types of technical debt in deep learning frameworks. *Empirical Software Engineering* 26 (2021), 1–36.

[122] Kristina Loncar, Jasmin Redzepagic, and Vedran Dakic. 2024. SECURE CODING GUIDELINES AND STANDARDS. *Annals of DAAAM & Proceedings* 35 (2024).

[123] Qinghua Lu, Yuxiu Luo, Liming Zhu, Mingjian Tang, Xiwei Xu, and Jon Whittle. 2023. Developing responsible chatbots for financial services: a pattern-oriented responsible artificial intelligence engineering approach. *IEEE Intelligent Systems* 38, 6 (2023), 42–51.

[124] Qinghua Lu, Liming Zhu, Xiwei Xu, and Jon Whittle. 2023. Responsible-AI-by-design: A pattern collection for designing responsible artificial intelligence systems. *Ieee Software* 40, 3 (2023), 63–71.

[125] Qinghua Lu, Liming Zhu, Xiwei Xu, Jon Whittle, David Douglas, and Conrad Sanderson. 2022. Software engineering for responsible AI: An empirical study and operationalised patterns. In *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice*. 241–242.

[126] Qinghua Lu, Liming Zhu, Xiwei Xu, Jon Whittle, Didar Zowghi, and Aurelie Jacquet. 2024. Responsible AI pattern catalogue: A collection of best practices for AI governance and engineering. *Comput. Surveys* 56, 7 (2024), 1–35.

[127] Majdi Maabreh, Omar Darwish, Ola Karajeh, and Yahya Tashtoush. 2022. On developing deep learning models with particle swarm optimization in the presence of poisoning attacks. In *2022 International Arab Conference on Information Technology (ACIT)*. IEEE, 1–5.

[128] Majdi Maabreh, Arwa Maabreh, Basheer Qolomany, and Ala Al-Fuqaha. 2022. The robustness of popular multiclass machine learning models against poisoning attacks: Lessons and insights. *International Journal of Distributed Sensor Networks* 18, 7 (2022), 15501329221105159.

[129] Alina Mailach and Norbert Siegmund. 2023. Socio-technical anti-patterns in building ML-enabled software: insights from leaders on the forefront. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 690–702.

[130] Silverio Martínez-Fernández, Justus Bogner, Xavier Franch, Marc Oriol, Julien Siebert, Adam Trendowicz, Anna Maria Vollmer, and Stefan Wagner. 2022. Software engineering for AI-based systems: a survey. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31, 2 (2022), 1–59.

[131] Silverio Martínez-Fernández, Justus Bogner, Xavier Franch, Marc Oriol, Julien Siebert, Adam Trendowicz, Anna Maria Vollmer, and Stefan Wagner. 2022. Software Engineering for AI-Based Systems: A Survey. *ACM Transactions on Software Engineering and Methodology* 31, 2 (2022). Cited by:

138; All Open Access, Green Open Access. doi:10.1145/3487043

[132] Wataru Matsuda, Mariko Fujimoto, Tomomi Aoyama, and Takuho Mitsunaga. 2019. Cyber security risk assessment on industry 4.0 using ics testbed with ai and cloud. In *2019 IEEE conference on application, information and network security (AINS)*. IEEE, 54–59.

[133] Mohcin Mekhfioui, Nabil El Bazi, Oussama Laayati, Amal Satif, Marouan Bouchouirbat, Chaïmaâ Kissi, Tarik Boujiha, and Ahmed Chebak. 2025. Optimized digital watermarking for robust information security in embedded systems. *Information* 16, 4 (2025), 322.

[134] Mark Huasong Meng, Guangdong Bai, Sin Gee Teo, Zhe Hou, Yan Xiao, Yun Lin, and Jin Song Dong. 2022. Adversarial Robustness of Deep Neural Networks: A Survey from a Formal Verification Perspective. *ArXiv* abs/2206.12227 (2022). https://api.semanticscholar.org/CorpusID:249223202

[135] Ahmed Menshawy, Zeeshan Nawaz, and Mahmoud Fahmy. 2024. Navigating Challenges and Technical Debt in Large Language Models Deployment. In *Proceedings of the 4th Workshop on Machine Learning and Systems*. 192–199.

[136] Faisal Mohammed. 2024. Developing Transparent AI Models to Enhance Interpretability and Trust in Medical Diagnostics: Implementing explainable AI techniques to provide transparent explanations for medical diagnoses, enhancing trust and acceptance among healthcare professionals. *Journal of Machine Learning for Healthcare Decision Support* 4, 2 (2024), 36–43.

[137] Vasilica-Andreea Moldovan, Liviu-Marian Berciu, and Rares-Danut Patcas. 2024. The python software quality dataset. In *2024 50th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 395–398.

[138] Sergio Moreschini, Valentina Lenarduzzi, and Ludovik Coba. 2024. Towards a Technical Debt for AI-based Recommender System. In *Proceedings of the 7th ACM/IEEE International Conference on Technical Debt*. 36–39.

[139] Mohan Vamsi Musunuru, Chiranjeevi Devi, and Swaminathan Sethuraman. 2025. Optimizing Hot Standby Redundancy Using AI for Network Traffic Balancing and Failover Management. *Journal of Knowledge Learning and Science Technology ISSN: 2959-6386 (online)* 4, 3 (2025), 14–26.

[140] Mallek Mziou Sallami, Mohamed Ibn Khedher, Asma Trabelsi, Samy Kerboua-Benlarbi, and Dimitri Bettebghor. 2019. Safety and robustness of deep neural networks object recognition under generic attacks. In *International conference on neural information processing*. Springer, 274–286.

[141] Muhammad Naeem, Tauseef Jamal, Jorge Diaz-Martinez, Shariq Aziz Butt, Nicolo Montesano, Muhammad Imran Tariq, Emiro De-la Hoz-Franco, and Ethel De-La-Hoz-Valdiris. 2021. Trends and future perspective challenges in big data. In *Advances in intelligent data analysis and applications: Proceeding of the sixth euro-China conference on intelligent data analysis and applications, 15–18 October 2019, Arad, Romania*. Springer, 309–325.

[142] Nadia Nahar, Shurui Zhou, Grace Lewis, and Christian Kästner. 2022. Collaboration challenges in building ml-enabled systems: Communication, documentation, engineering, and process. In *Proceedings of the 44th international conference on software engineering*. 413–425.

[143] MK Nallakaruppan, Balamurugan Balusamy, M Lawanya Shri, V Malathi, and Siddhartha Bhattacharyya. 2024. An Explainable AI framework for credit evaluation and analysis. *Applied Soft Computing* 153 (2024), 111307.

[144] Mary Nankya, Allan Mugisa, Yusuf Usman, Aadesh Upadhyay, and Robin Chataut. 2024. Security and Privacy in E-Health Systems: A Review of AI and Machine Learning Techniques. *IEEE Access* (2024).

[145] Sidhant Narula, Mohammad Ghasemigol, Javier Carnerero-Cano, Amanda Minnich, Emil Lupu, and Daniel Takabi. 2025. Exploring AI Security: A Systematic Mapping Study. *IEEE Access* (2025).

[146] Amin Nikanjam and Foutse Khomh. 2021. Design smells in Deep Learning programs: an empirical study. In *2021 IEEE International conference on software maintenance and evolution (ICSME)*. IEEE, 332–342.

[147] NIST. 2024. AI Risks and Trustworthiness. *https://airc.nist.gov/airmf-resources/airmf/3-sec-characteristics/* (2024).

[148] David OBrien, Sumon Biswas, Sayem Imtiaz, Rabe Abdalkareem, Emad Shihab, and Hridesh Rajan. 2022. 23 shades of self-admitted technical debt: An empirical study on machine learning software. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 734–746.

[149] Abayomi Titilola Olutimehin, Adekunbi Justina Ajayi, Olufunke Cynthia Metibemu, Adebayo Yusuf Balogun, Tunboson Oyewale Oladoyinbo, and Oluwaseun Oladeji Olaniyi. 2025. Adversarial threats to AI-driven systems: Exploring the attack surface of machine learning models and countermeasures. *Available at SSRN 5137026* (2025).

[150] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems* 35 (2022), 27730–27744.

[151] Pavan Paidy and Krishna Chaganti. 2024. Securing AI-driven APIs: Authentication and abuse prevention. *International Journal of Emerging Research in Engineering and Technology* 5, 1 (2024), 27–37.

[152] Gaurav Parmar, Rimi Gupta, Tejas Bhatt, GJ Sahani, Brijeshkumar Y Panchal, and Hiren Patel. 2023. A review on data balancing techniques and machine learning methods. In *2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT)*. IEEE, 1004–1008.

[153] Jay Patel and Harshal Shah. 2021. Creating Safe and Secure AI-From Computer Design to Cloud Technology. *INTERNATIONAL RESEARCH JOURNAL OF ENGINEERING & APPLIED SCIENCES* 9, 4 (2021), 10–55083.

[154] Dino Pedreschi, Fosca Giannotti, Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, and Franco Turini. 2019. Meaningful explanations of black box AI decision systems. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 9780–9784.

[155] Boris Pérez, Camilo Castellanos, Darío Correal, Nicolli Rios, Sávio Freire, Rodrigo Spínola, Carolyn Seaman, and Clemente Izurieta. 2021. Technical debt payment and prevention through the lenses of software architects. *Information and Software Technology* 140 (2021), 106692.

[156] Boris Pérez, Darío Correal, and Hernán Astudillo. 2019. A proposed model-driven approach to manage architectural technical debt life cycle. In *2019 IEEE/ACM International Conference on Technical Debt (TechDebt)*. IEEE, 73–77.

[157] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. 2008. Systematic mapping studies in software engineering. In *12th international conference on evaluation and assessment in software engineering (EASE)*. BCS Learning & Development.

[158] Catherine Petrozzino. 2021. Who pays for ethical debt in AI? *AI and Ethics* 1, 3 (2021), 205–208.

[159] Neoklis Polyzotis, Sudip Roy, Steven Euijong Whang, and Martin Zinkevich. 2018. Data lifecycle challenges in production machine learning: a survey. *ACM SIGMOD Record* 47, 2 (2018), 17–28.

[160] Mitra Pooyandeh, Ki-Jin Han, and Insoo Sohn. 2022. Cybersecurity in the AI-Based metaverse: A survey. *Applied Sciences* 12, 24 (2022), 12993.

[161] Gopi Krishnan Rajbahadur, Keheliya Gallaba, Elyas Rashno, Arthit Suriyawongkul, Karen Bennet, Kate Stewart, and Ahmed E Hassan. 2025. Building an Open AIBOM Standard in the Wild. *arXiv preprint arXiv:2510.07070* (2025).

[162] Prajit T. Rajendran, Huascar Espinoza, Agnes Delaborde, and Chokri Mraidha. 2021. Human-in-the-Loop Learning Methods Toward Safe DL-Based Autonomous Systems: A Review. In *Computer Safety, Reliability, and Security. SAFECOMP 2021 Workshops: DECSoS, MAPSOD, DepDevOps, USDAI, and WAISE, York, UK, September 7, 2021, Proceedings* (York, United Kingdom). Springer-Verlag, Berlin, Heidelberg, 251–264. doi:10.1007/978-3-030-83906-2_20

[163] Legha Mamta Ranjitsingh and TV Subba Rao. 2025. Establish legal and regulatory standards for the testing and validation of AI systems to ensure their reliability and safety in operational environments. *International Journal of System Assurance Engineering and Management* 16, 10 (2025), 3338–3353.

[164] Gilberto Recupito, Fabiano Pecorelli, Gemma Catolino, Valentina Lenarduzzi, Davide Taibi, Dario Di Nucci, and Fabio Palomba. 2024. 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.

[165] Gilberto Recupito, Raimondo Rapacciuolo, Dario Di Nucci, and Fabio Palomba. 2024. Unmasking data secrets: An empirical investigation into data smells and their impact on data quality. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*. 53–63.

[166] Xavier Renard, Thibault Laugel, and Marcin Detyniecki. 2024. Understanding prediction discrepancies in classification. *Machine Learning* 113, 10 (2024), 7997–8026.

[167] Roberto Riggio, Estefanía Coronado, Neiva Linder, Adzic Jovanka, Gianpiero Mastinu, Leonardo Goratti, Miguel Rosa, Hans Schotten, and Marco Pistore. 2021. Ai@ edge: A secure and reusable artificial intelligence platform for edge computing. In *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. IEEE, 610–615.

[168] Drew Roselli, Jeanna Matthews, and Nisha Talagala. 2019. Managing bias in AI. In *Companion proceedings of the 2019 world wide web conference*. 539–544.

[169] Giulio Rossolini, Alessandro Biondi, and Giorgio Buttazzo. 2022. Increasing the Confidence of Deep Neural Networks by Coverage Analysis. 802 – 815 pages.

[170] Christoph Ruland and Jochen Sassmannshausen. 2018. Access control in safety critical environments. In *2018 12th International Conference on Reliability, Maintainability, and Safety (ICRMS)*. IEEE, 223–229.

[171] Wissam Salhab, Darine Ameyed, Fehmi Jaafar, and Hamid Mcheick. 2024. A systematic literature review on ai safety: Identifying trends, challenges and future directions. *IEEE Access* (2024).

[172] Amir Samadi, Amir Shirian, Konstantinos Koufos, Kurt Debattista, and Mehrdad Dianati. 2023. SAFE: Saliency-aware counterfactual explanations for DNN-based automated driving systems. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 5655–5662.

[173] Olusoji John Samuel. 2025. Adversarial AI the New Frontier in Cybersecurity Threats and Defenses. *Journal of Science, Technology and Engineering Research* 3, 1 (2025), 1–13.

[174] Iqbal H Sarker. 2024. Introduction to AI-driven cybersecurity and threat intelligence. In *AI-driven cybersecurity and threat intelligence: Cyber automation, intelligent decision-making and explainability*. Springer, 3–19.

[175] Darius Sas and Paris Avgeriou. 2023. An architectural technical debt index based on machine learning and architectural smells. *IEEE Transactions on Software Engineering* 49, 8 (2023), 4169–4195.

[176] Simon Schneider, Ananya Saha, Emanuele Mezzi, Katja Tuma, and Riccardo Scandariato. 2024. Designing Secure AI-based Systems: a Multi-Vocal Literature Review. In *2024 IEEE Secure Development Conference (SecDev)*. IEEE, 13–19.

[177] Daniel Schwartz, Yigit Alparslan, and Edward Kim. 2020. Regularization and sparsity for adversarial robustness and stable attribution. In *Advances in Visual Computing: 15th International Symposium, ISVC 2020, San Diego, CA, USA, October 5–7, 2020, Proceedings, Part I 15*. Springer, 3–14.

[178] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in machine learning systems. *Advances in neural information processing systems* 28 (2015).

[179] Osama R Shahin, Hamoud H Alshammari, Raed N Alabdali, Ahmed M Salaheldin, and Neven Saleh. 2025. Automated multi-model framework for malaria detection using deep learning and feature fusion. *Scientific Reports* 15, 1 (2025), 25672.

[180] Sakib Shahriar, Sonal Allana, Seyed Mehdi Hazratifard, and Rozita Dara. 2023. A survey of privacy risks and mitigation strategies in the Artificial Intelligence life cycle. *IEEE Access* 11 (2023), 61829–61854.

[181] Yonadav Shavit, Sandhini Agarwal, Miles Brundage, Steven Adler, Cullen O'Keefe, Rosie Campbell, Teddy Lee, Pamela Mishkin, Tyna Eloundou, Alan Hickey, et al. 2023. Practices for governing agentic AI systems. *Research Paper, OpenAI, December* (2023).

[182] R Sheeba, Jay Prakash Mahto, Syed Sabith Ansari, Zian Rajeshkumar Surani, P Chinnasamy, and Manjunathan Alagarsamy. 2025. Decentralized Data Validation for Ethical AI Training. In *2025 International Conference on Computational Robotics, Testing and Engineering Evaluation (ICCRTEE)*.

IEEE, 1–6.

[183] Raymond Sheh. 2021. Explainable artificial intelligence requirements for safe, intelligent robots. In *2021 IEEE international conference on intelligence and safety for robotics (ISR)*. IEEE, 382–387.

[184] Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. 2022. Back to the Drawing Board: A Critical Evaluation of Poisoning Attacks on Production Federated Learning. *Proceedings - IEEE Symposium on Security and Privacy* 2022-May (2022), 1354 – 1371. Cited by: 192; All Open Access, Green Open Access. doi:10.1109/SP46214.2022.9833647

[185] Yuxin Shi, Han Yu, and Cyril Leung. 2023. Towards fairness-aware federated learning. *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[186] Karthik Shivashankar and Antonio Martini. 2022. Maintainability challenges in ML: A systematic literature review. In *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 60–67.

[187] Ben Shneiderman. 2020. Bridging the gap between ethics and practice: guidelines for reliable, safe, and trustworthy human-centered AI systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 10, 4 (2020), 1–31.

[188] Arumoy Shome, Luis Cruz, and Arie Van Deursen. 2022. Data smells in public datasets. In *Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI*. 205–216.

[189] Raj Mani Shukla and John Cartlidge. 2022. 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, 0119–0124.

[190] Emmanuel Iko-Ojo Simon, Melina Vidoni, and Fatemeh H Fard. 2023. Algorithm Debt: Challenges and Future Paths. In *2023 IEEE/ACM 2nd International Conference on AI Engineering–Software Engineering for AI (CAIN)*. IEEE, 90–91.

[191] Alessandro Simonetta and Maria Cristina Paoletti. 2024. ISO/IEC Standards and Design of an Artificial Intelligence System. (2024).

[192] Prabath Siriwardena. 2019. *Advanced API security: OAuth 2.0 and beyond*. Apress.

[193] Dionysios Sklavenitis and Dimitris Kalles. 2024. Measuring Technical Debt in AI-Based Competition Platforms. In *Proceedings of the 13th Hellenic Conference on Artificial Intelligence*. 1–10.

[194] Kacper Sokol and Peter Flach. 2019. Counterfactual explanations of machine learning predictions: opportunities and challenges for AI safety. In *2019 AAAI Workshop on Artificial Intelligence Safety, SafeAI 2019*. CEUR Workshop Proceedings.

[195] Aditya K Sood and Sherali Zeadally. 2025. Malicious AI Models Undermine Software Supply-Chain Security. *Commun. ACM* 68, 6 (2025), 62–71.

[196] Balaji Soundararajan. [n. d.]. Secure Configuration Management for Microservices Architecture. ([n. d.]).

[197] Petr Spelda and Vit Stritecky. 2025. Security practices in AI development. *AI & SOCIETY* (2025), 1–11.

[198] K. G. Srinivasa, Muralidhar Kurni, and Kuppala Saritha. 2022. *Harnessing the Power of AI to Education*. Springer Nature Singapore, Singapore, 311–342. doi:10.1007/978-981-19-6734-4_13

[199] Ramya Srinivasan and Ajay Chander. 2019. Understanding Bias in Datasets using Topological Data Analysis.. In *AISafety@ IJCAI*.

[200] André Steimers and Thomas Bömer. 2021. Sources of risk and design principles of trustworthy artificial intelligence. In *International Conference on Human-Computer Interaction*. Springer, 239–251.

[201] Mark A Sujan. 2023. Looking at the Safety of AI from a Systems Perspective: Two Healthcare Examples. In *Safety in the Digital Age: Sociotechnical Perspectives on Algorithms and Machine Learning*. Springer Nature Switzerland Cham, 79–90.

[202] B. Sujatha, K. Anas Faraz, N. Pranathi, Ch. B. R. Saranya, and B.S.V. Chaitanya. 2023. Securing data with blockchain and AI. *AIP Conference Proceedings* 2492 (2023). Cited by: 1. doi:10.1063/5.0115385

[203] Edi Sutoyo and Andrea Capiluppi. 2024. SATDAUG-A Balanced and Augmented Dataset for Detecting Self-Admitted Technical Debt. In *Proceedings of the 21st International Conference on Mining Software Repositories*. 289–293.

[204] Yiming Tang, Raffi Khatchadourian, Mehdi Bagherzadeh, Rhia Singh, Ajani Stewart, and Anita Raja. 2021. An empirical study of refactorings and technical debt in machine learning systems. In *2021 IEEE/ACM 43rd international conference on software engineering (ICSE)*. IEEE, 238–250.

[205] Bilel Tarchoun, Anouar Ben Khalifa, and Mohamed Ali Mahjoub. 2022. Investigating the robustness of multi-view detection to current adversarial patch threats. In *2022 6th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*. IEEE, 1–6.

[206] Chandu Thota, Revathi Sundarasekar, Gunasekaran Manogaran, R Varatharajan, and MK Priyan. 2018. Centralized fog computing security platform for IoT and cloud in healthcare system. In *Fog computing: Breakthroughs in research and practice*. IGI global, 365–378.

[207] Haileleol Tibebu. 2024. Framework for Data Protection, Security, and Privacy in AI Applications. *The Broadcast Centre Here East, London* (2024).

[208] Andrea C Tricco, Erin Lillie, Wasifa Zarin, Kelly K O'Brien, Heather Colquhoun, Danielle Levac, David Moher, Micah DJ Peters, Tanya Horsley, Laura Weeks, et al. 2018. PRISMA extension for scoping reviews (PRISMA-ScR): checklist and explanation. *Annals of internal medicine* 169, 7 (2018), 467–473. doi:10.7326/M18-0850

[209] Bekir Tolga Tutuncuoglu. 2024. Zero-Downtime AI: Predictive and Autonomous Server Restoration Without Human Input. *Available at SSRN 5249062* (2024).

[210] Bart Van Oort, Luís Cruz, Maurício Aniche, and Arie Van Deursen. 2021. The prevalence of code smells in machine learning projects. In *2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)*. IEEE, 1–8.

[211] Mathew J Walter, Aaron Barrett, and Kimberly Tam. 2024. A red teaming framework for securing AI in maritime autonomous systems. *Applied Artificial Intelligence* 38, 1 (2024), 2395750.

[212] Hao Wang, Chen Li, Jinzhe Jiang, Xin Zhang, Yaqian Zhao, and Weifeng Gong. 2023. Distribution-restrained Softmax Loss for the Model Robustness. *arXiv preprint arXiv:2303.12363* (2023).

[213] Xiaofei Wang, Herbert Schuster, Reuben Borrison, and Benjamin Kløpper. 2023. Technical Debt Management in Industrial ML-State of Practice and Management Model Proposal. In *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*. IEEE, 1–9.

[214] Xinda Wang, Shu Wang, Pengbin Feng, Kun Sun, Sushil Jajodia, Sanae Benchaaboun, and Frank Geck. 2021. Patchrnn: A deep learning-based system for security patch identification. In *MILCOM 2021-2021 IEEE Military Communications Conference (MILCOM)*. IEEE, 595–600.

[215] Hironori Washizaki, Hiromu Uchida, Foutse Khomh, and Yann-Gael Gueheneuc. 2019. Studying software engineering patterns for designing machine learning systems. In *2019 10th International Workshop on Empirical Software Engineering in Practice (IWESEP)*. IEEE, 49–495.

[216] Benjamin D Werner, Benjamin J Schumeg, Tiffany M Mills, and Elizabeth V Velilla. 2023. An Assurance Case for the DoD Ethical Principles of Artificial Intelligence. In *2023 Annual Reliability and Maintainability Symposium (RAMS)*. IEEE, 1–7.

[217] Claes Wohlin. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*. 1–10.

[218] Boming Xia. 2025. *Operationalising Safe and Responsible AI: A System Level Perspective*. Ph. D. Dissertation. UNSW Sydney.

[219] Boming Xia, Tingting Bi, Zhenchang Xing, Qinghua Lu, and Liming Zhu. 2023. An Empirical Study on Software Bill of Materials: Where We Stand and the Road Ahead. *Proceedings - International Conference on Software Engineering* (2023), 2630 – 2642. Cited by: 37; All Open Access, Green Open Access. doi:10.1109/ICSE48619.2023.00219

[220] Rodrigo Ximenes, Antonio Pedro Santos Alves, Tatiana Escovedo, Rodrigo Spinola, and Marcos Kalinowski. 2025. Investigating Issues that Lead to Code Technical Debt in Machine Learning Systems. In *2025 IEEE/ACM 4th International Conference on AI Engineering–Software Engineering for AI (CAIN)*. IEEE, 173–183.

[221] Feiyu Xu, Hans Uszkoreit, Yangzhou Du, Wei Fan, Dongyan Zhao, and Jun Zhu. 2019. Explainable AI: A Brief Survey on History, Research Areas, Approaches and Challenges. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11839 LNAI (2019), 563 – 574. Cited by: 486. doi:10.1007/978-3-030-32236-6_51

[222] Meng Yan, Xin Xia, Emad Shihab, David Lo, Jianwei Yin, and Xiaohu Yang. 2018. Automating change-level self-admitted technical debt determination. *IEEE Transactions on Software Engineering* 45, 12 (2018), 1211–1229.

[223] Haoran Yang, Yu Nong, Shaowei Wang, and Haipeng Cai. 2024. Multi-language software development: Issues, challenges, and solutions. *IEEE Transactions on Software Engineering* 50, 3 (2024), 512–533.

[224] Haiyin Zhang, Luís Cruz, and Arie Van Deursen. 2022. Code smells for machine learning applications. In *Proceedings of the 1st international conference on AI engineering: software engineering for AI*. 217–228.

[225] Kaiyue Zhang, Xuan Song, Chenhan Zhang, and Shui Yu. 2022. Challenges and future directions of secure federated learning: a survey. *Frontiers of Computer Science* 16, 5 (2022). Cited by: 94; All Open Access, Bronze Open Access, Green Open Access. doi:10.1007/s11704-021-0598-z

[226] Weimin Zhao, Sanaa Alwidian, and Qusay H. Mahmoud. 2022. Adversarial Training Methods for Deep Learning: A Systematic Review. *Algorithms* 15, 8 (2022). doi:10.3390/a15080283

[227] Xingyu Zhao, Alec Banks, James Sharp, Valentin Robu, David Flynn, Michael Fisher, and Xiaowei Huang. 2020. A safety framework for critical systems utilising deep neural networks. In *Computer Safety, Reliability, and Security: 39th International Conference, SAFECOMP 2020, Lisbon, Portugal, September 16–18, 2020, Proceedings 39*. Springer, 244–259.

[228] Xingyu Zhao, Wei Huang, Sven Schewe, Yi Dong, and Xiaowei Huang. 2021. Detecting operational adversarial examples for reliable deep learning. In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*. IEEE, 5–6.

[229] Lanyue Zhi, Shaoguang Liu, Haoqi Dai, Mingwei Liu, and Jinhe Wang. 2024. Algorithm for Data Format Conversion and Compatibility Guarantee in Technical Platforms for Cross Platform Application Integration. In *2024 IEEE 4th International Conference on Data Science and Computer Application (ICDSCA)*. IEEE, 893–897.

[230] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. 2021. A Comprehensive Survey on Transfer Learning. *Proc. IEEE* 109, 1 (2021), 43–76. doi:10.1109/JPROC.2020.3004555

[231] Daniel M. Ziegler, Seraphina Nix, Lawrence Chan, Tim Bauman, Peter Schmidt-Nielsen, Tao Lin, Adam Scherlis, Noa Nabeshima, Ben Weinstein-Raun, Daniel de Haas, Buck Shlegeris, and Nate Thomas. 2022. Adversarial training for high-stakes reliability. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '22)*. Curran Associates Inc., Red Hook, NY, USA, Article 674, 13 pages.

Received 25 July 2026; revised ; accepted